Universidade de Aveiro

2018

Departamento de Eletrónica, Telecomunicações e Informática

João Carlos Costa Abrunhosa

diversidade.ID – Projeto de Ciência Cidadã para Apoiar a Observação de Insectos

diversidade.ID – Citizen Science Project to Help the Observation of Insects

Universidade de Aveiro

2018

Departamento de Eletrónica, Telecomunicações e Informática

João Carlos Costa Abrunhosa

diversidade.ID – Projeto de Ciência Cidadã para Apoiar a Observação de Insectos

diversidade.ID – Citizen Science Project to Help the Observation of Insects

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor Ilídio Castro Oliveira, Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e da Doutora Olga Ameixa, investigador da Unidade CESAM da Universidade de Aveiro

Dedico todo este trabalho à minha família que sempre me apoiou e incentivou, a todos os meus amigos que de certa maneira me ajudaram no desenvolvimento e a todo o apoio incondicional.

o júri

presidente	Professor Doutor Paulo Miguel de Jesus Dias, Professor Auxiliar, Universidade de Aveiro
vogais	Professor Doutor Pedro Miguel dos Santos Beça Pereira, Professor Auxiliar, Universidade de Aveiro
	Professor Doutor Ilídio Fernando de Castro Oliveira, Professor Auxiliar, Universidade de Aveiro

agradecimentos

Quero agradecer aos meus orientadores, Ilídio Oliveira e Olga Ameixa, pelo constante apoio, paciência e conhecimento transmitido ao longo do desenvolvimento da dissertação. Agradeço também a todos que, de alguma maneira, contribuíram para o desenvolvimento, especialmente a minha família pelo seu incansável apoio.

palavras-chave

Entomologia, Ciência Cidadã, Capturas, Computação Móvel, Desenvolvimento Web.

resumo

O envolvimento dos cidadãos na observação da natureza é bastante promovido por várias iniciativas relacionadas à biodiversidade. Para além do envolvimento de cidadãos, há às vezes o benefício de coleção distribuída de dados, ao oferecer ferramentas que podem ser usadas em dispositivos móveis para captura e documentação dessas observações.

Entomologia é um dos campos de estudo em que o uso de observações reportadas por utilizadores, por curiosos ou amantes da natureza, pode ajudar na monitorização da população de insetos, especificamente em certos ecossistemas, como o Baixo Vouga Lagunar.

Neste trabalho, nós propomos uma aplicação móvel (diversidade.ID) que possibilita qualquer cidadão de reportar observações de insetos. O avistamento é expectável que seja acompanhado por fotos e dados de localização, adquiridos pelo telemóvel. As observações distribuídas são agregadas para posterior avaliação por colaboradores, usando um sistema de gerenciamento de listas de dados. Depois da classificação, as observações são partilhadas na plataforma para acesso geral e exploração.

A aplicação diversidade.ID está disponível para sistemas Android e tem sido usada de forma exploratória por estudantes.

keywords Entomology, Citizen Science, Captures, Mobile Computing, Web Development.

abstract

The citizen involvement in nature observation is being promoted by several biodiversity iniatives. Beyond the citizen engagement, there is sometimes the benefit of distributed data collection, by offering tools that can be used in mobile devices for capturing and document observations.
Entomology is one of those fields in which the use of self-reported observations by curious and nature-lovers can help monitoring the populations of insects, especially in specific ecossystems, like the Baixo Vouga Lagunar in Aveiro.
In this work, we propose a mobile app (diversidade.ID) that enables any citizen to report the observation of insects. The sighting is expected to be supplemented with photos and location data, acquired with the mobile phone. The distributed observations are aggregated for posterior review by collaborators, using a backoffice worklists management system. After classification, observations and shared in the platform for general access and exploration.
The diversidade.ID application is available for the Android system and has been used in exploratory mode from students.

Contents

List of Figures

List of Tables

Glossary

SDK	Software Development Kit
API	Application Programming Interface
ADT	Android Development Tools
OS	Operating System
CMS	Content Management System
UI	User Interface
FTP	File Transfer Protocol
URL	Uniform Resource Locator
GPS	Global Positioning System
UID	Unique Identifier
HTTP	HyperText Transfer Protocol
URI	Uniform Resource Identifier
SUS	System Usability Scale
UK	United Kingdom

1 Introduction

1.1 Motivation

Entomologists can be contributors to the betterment of humankind by studying the role of the insects in areas like pest control, spreading of diseases, and damage to crops and animals. They also can study how beneficial insects can be to the world.

There's an interest in opening science to citizen involvement, with divulgation and sharing initiatives, making transparent the development and research done. This initiative invites the common user with the sharing of captures and their information, helping entomologists with more quantity and diverse data [1]. This can be a great opportunity for nature-lovers, as well as completely new to the field citizens, to have their opportunity to help the development of entomology, accelerating the process of information gathering and consequent documentation [1] [2].

Since this project has the focus of the city of Aveiro and its surroundings, the Baixo Vouga Lagunar zone stands out as an ecosystem favorable to the breeding and discovery of insects.

The platform diversidade.ID is a great way to help entomologists with the documentation of species, by using the common citizen with a smartphone a tool [1]. Everyone can use this platform as a way of contributing to science, while having fun doing so.

1.2 Objectives

The main objective of this Project is to design and implement a computing platform that supports the capturing and documentation of a variety of different insects' species that belong and co-habit on Aveiro's Ria, in Baixo Vouga Lagunar. These distributed captures are reviewed by collaborators, users that document the details of these captures. Those captures are then publicly shared to the benefit of the informal entomology community.

This is a citizen and science gathering effort, helping entomologists document with ease the various insects that exist.

There are two main platforms: mobile application and website.

The mobile application is made with capturing as the main objective, using some capabilities from the smartphone, such as GPS, to capture the capture location. All other features can be done on the mobile application as well, such as: check other user's captures, species information, species location,

The website can do the same thing as the mobile application but with an extra: blog. The blog is only managed by the administrator, where he can post news, accept or deny commentaries. The administrator can also change the look and functions of the website.

1.3 Main contributions

This platform has the goal to deliver a working platform that helps with the capture and documentation of insect species located in Baixo Vouga Lagunar. With the help of citizens and their mobile handsets, the entomologists can have a great number of contributions that facilitates the development and research of insect species.

By the end of this project development and dissertation, the following have been accomplished:

- Mobile Application (Android) for insects capture and documentation – open source
- Web Application for insects capture and documentation – open source
- Poster - J.Abrunhosa, 2018, “diversidadeID: uma aplicação móvel para apoiar a ciência cidadã em entomologia”, Students@DETI, Universidade de Aveiro

1.4 Document structure

This document will describe in detail the creation of this platform, from the prototyping and design to its implementation.

Initially, all the technology used in this platform will be explained, and why they were chosen in the first place, backed with references. It will also be shown the already existing related projects in this field, and how they compare to diversidade.ID. This is will be done in the Chapter 2 of this document.

In Chapter 3, the use cases will be described. These use cases will describe the basic functionalities that the platform should have to meet the goals initially proposed. Those use cases are detailed, and they are backed with functional stories, that show the flow of the system and the expected behavior.

Next, in Chapter 4, the prototyping and initials designs will be explained, showing the choices made and even some early concepts of how the platform would be designed. This will go into detail on all choices made.

Chapter 5 will describe the implementation. The explanation of the implementation is extensively described and even explains the support technologies used in conjunction with the ones already explained in Chapter 2.

In Chapter 6, results and validation of the platform are discussed. All tests made, including usability tests and pilot use scenario, are described in this chapter and their results shown.

Chapter 7 will talk about the conclusion and what was achieved with this project. It will also include what can be done in the future, to further develop what already has been made.

2 State of the art

User reported events are used as a way to facilitate the work of gathering and documenting information. They use people without any background from the field to help the gathering of information. This can be quite advantageous, regarding the field of study [3].

By having many users gathering information for them, specialists will have a much higher quantity of information to deal with. By using their smartphones or computers, these users have the ability and power to help science, "they are part of a group of 'citizen scientists', networks of non-scientists to help to analyse and collect data as part of a research-led field" [1]. This will increase the odds of a particular research turn into a "legitimate, publishable research project" [1]. But this also can mean that some problems can come from it, like wrong information capture or documentation – "scientists have to ensure that the data is sound" [1]. Even with the negative aspects to it, user reported events can prove to be valuable if correctly used and can accelerate the process of information gathering and consequent advancement on research in various fields of study. This is notorious in the variety of already existing user-reported events applications.

Not only it helps the data gathering but it also helps the contributors to gain or improve their attitude towards insects and nature overall. Contributory entomology citizen science projects "affects participants' science self-efficacy, self-efficacy for environmental action, nature relatedness and attitude towards insects" [4].

Citizen science initiatives can also be seen as educational science, strengthening participant's understanding of science, environment and thinking skills regarding this matter. Clearly, citizen science projects are a great way to "passive accumulate data from enthusiasts, experts, and amateurs" [3]. As an entomologist "you get the information you need at the same time that you are getting people involved" [1].

2.1 Related work

2.1.1 Platforms for user-reported events in biology sciences

There are a variety of already established platforms for user-reported events. Some of them use mobile applications for a way to capture insects, others work only through a website. Most of them are not exactly made for a variety of species – for example, one is for stag beetles, while other is only made for mosquitos. Not only that, but their goal is different.

On this chapter, we will see some of those applications, what they do, and how they compare to the one on this dissertation.

Vacaloura

Vacaloura is a project that works in the same way as diversidade.ID – it uses citizens as a way to capture and document stag beetles. It has a partnership with the Department of

Wild Life of Biology Department in University of Aveiro. The objective of this project is to compile and organize information sent by users about the distribution of stag beetles and related from the Lucanidae family in the Portuguese territory. It also contributes to the European Network Monitoring of Stag Beetles, program that looks to evaluate the current situation and distribution of the specie [3].

This project is based only on a web application that lets the users upload the sightings of any stag beetle on the Portuguese territory. The following picture shows an example of the map distribution of this specie:

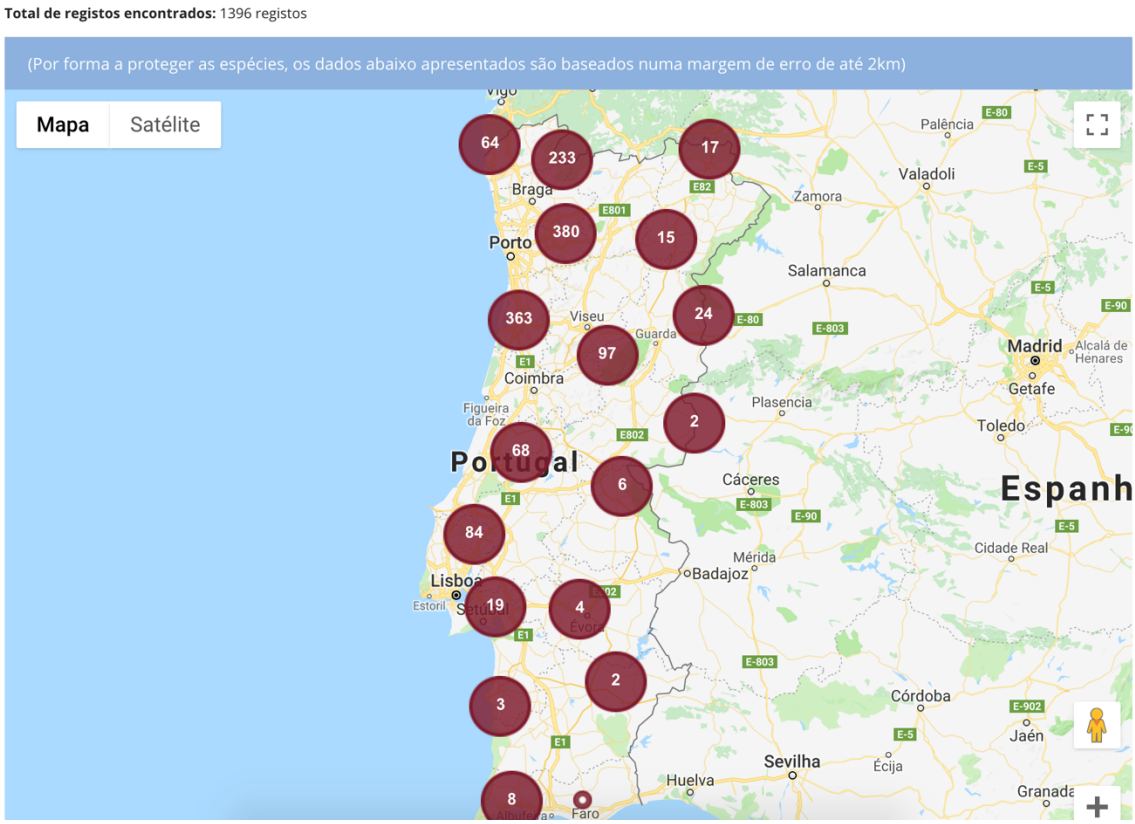

Figure 1 - Stag Beetle distribution in Portuguese territory (Vacaloura project)

It shows the number of recordings done and shows the number of captures per region. Clicking on a region shows the following:

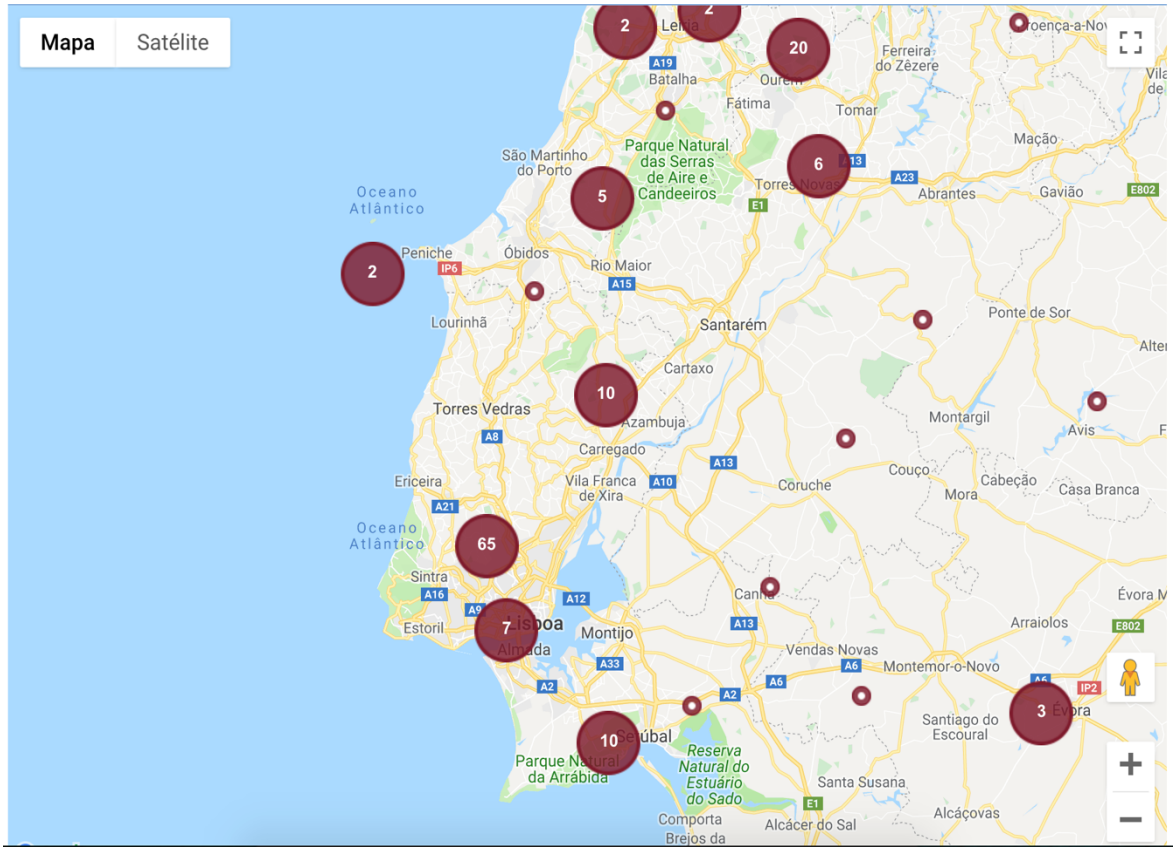

Figure 2 - Stag Beetle distribution in Portuguese territory (Vacaloura project)

This picture shows the distribution inside a region and the number of the recordings. By clicking on an individual capture, it shows the gender of the insect and the date of when it was taken.

As already stated, this application goal is only the stag beetle distribution in the Portuguese territory.

MozzWear

MozzWear is a platform designed to track mosquitos through sound capture. It records mosquitos sounds from up to 10 centimeters away. This application runs on Android and iOS, and it works solely by using machine learning to compare the sounds captured to the ones already documented. The data labels are obtained through a collective data tagging done by using a platform called Zooniverse. This platform is a large citizen science platform that gathers all kinds of information, from art and languages to biology [4]. The main goal of this platform is to recognize mosquitos that usually carry malaria with them. This application will be deployed in areas where this disease is still common, like developing countries [5].

Not only this application lets the user take captures of mosquitos, it also helps the user detect symptoms by giving it a simple Symptom Test. This is let the user know about the malaria disease and helps monitor it.

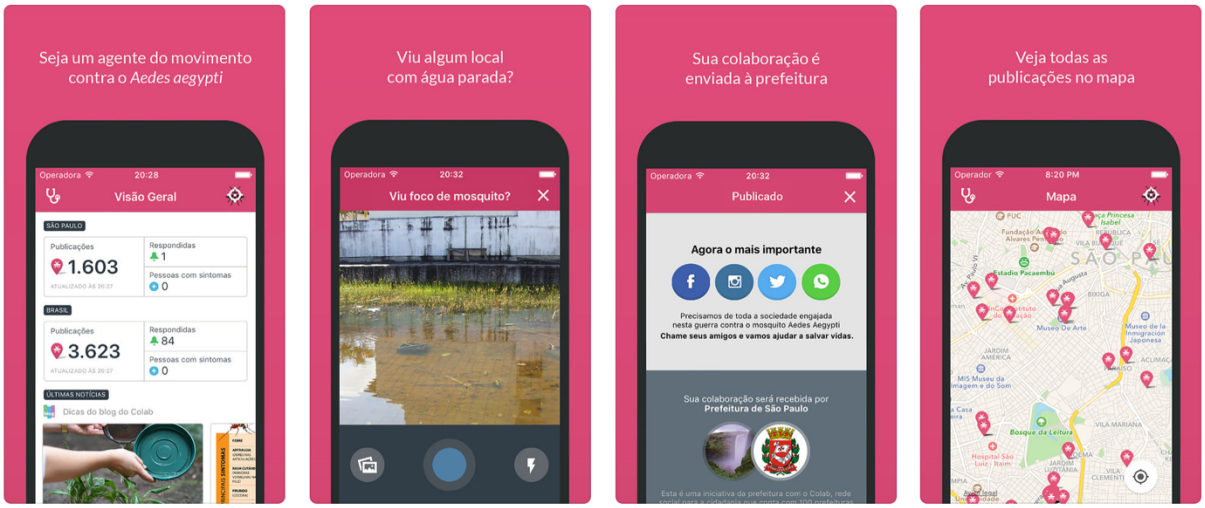

Figure 3 - Mozzwear application screens

The main focus of the application is the *Aeges aegypti* specie, spreader of the dengue fever, and it helps fight the spreading by capturing the position of the specie, controlling it.

iRecord

iRecord can be considered the application that is closer to what diversidade.ID wants to accomplish. The difference is that iRecord is an application that works for the UK territory, while diversidade.ID has the main goal the Baixo Vouga Lagunar zone, even though it can be deployed anywhere in the world.

The goal of iRecord is to collect and evaluate user captures. Users can share their captures with another members, explore the maps and it also offers graphs with the collected user data. This application is made for wildlife in general, not only insects. This makes it a more general application, with no real focus on one specie or animal group [6].

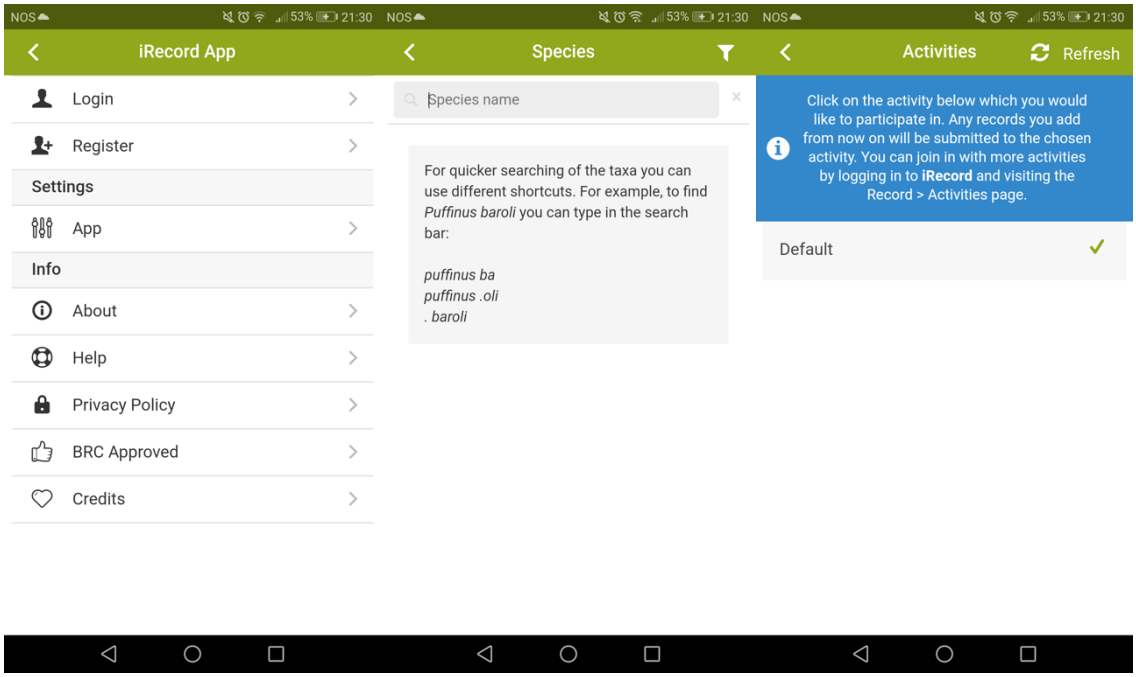

Figure 4 - iRecord screens

As seen in the image above (Figure 4), some of the actions on this application include the search for species and activities. These activities are gatherings of captures made specifically for this activity, like a group.

iNaturalist

iNaturalist is an application that helps users identify plants and animals around them. There are over 750,000 users using this application right now. It also works like diversidade.ID, letting users record their findings and share them with the community [7].

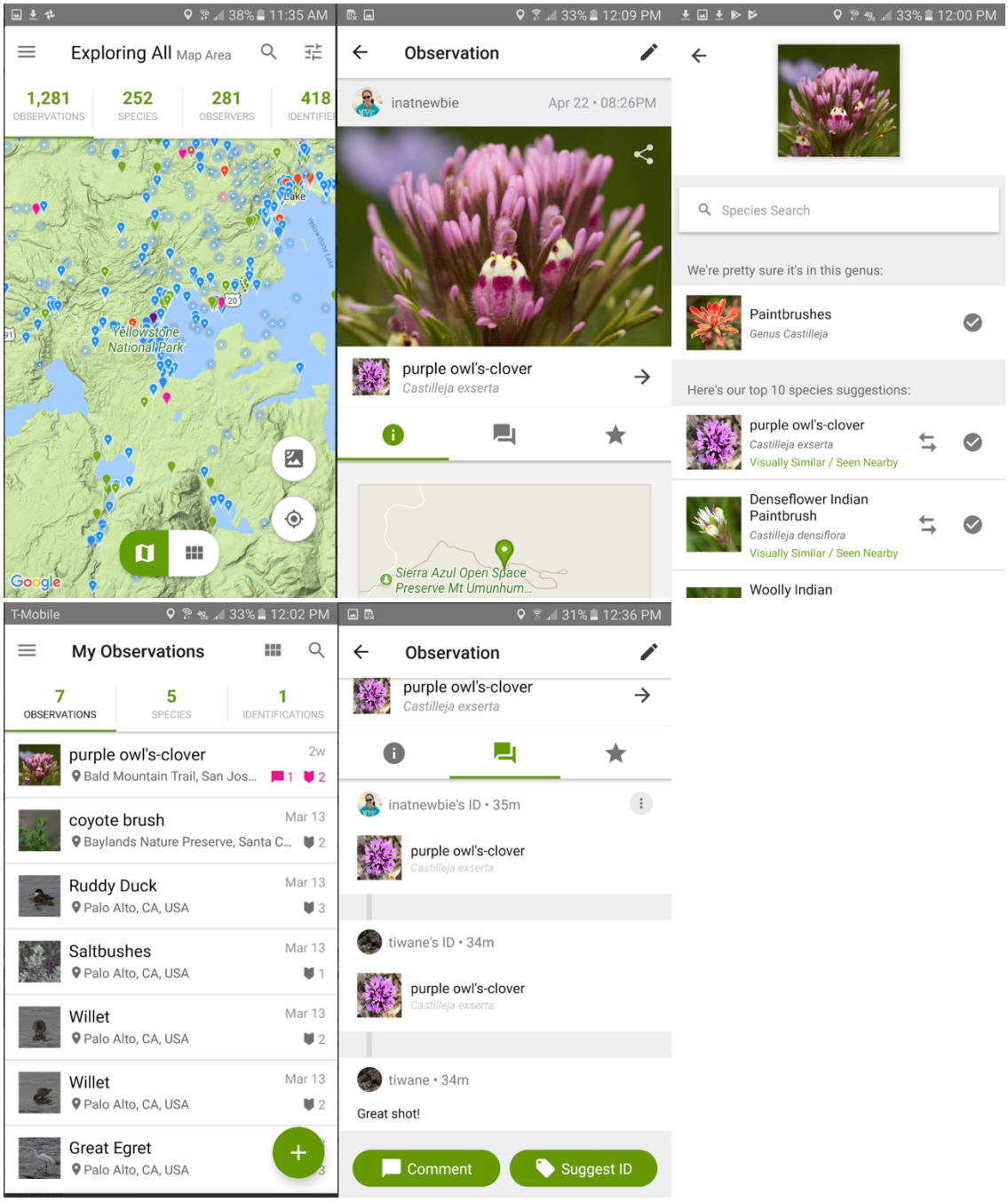

Figure 5 - Screens of the iNaturalist application

As seen in the images, it's possible to map the location of the captures, see their details, check the user's observations and its comments.

Mammal Mapper

This application, as the name implies, is a software that maps mammals around the area. Mammal mapper has the objective of making it easier and accessible to monitor and map mammal species. Whether the user is walking or riding a bicycle, they can capture the location of animals they see during those activities. This application also has a guide that helps the users identify the animals seen through their appearance [8].

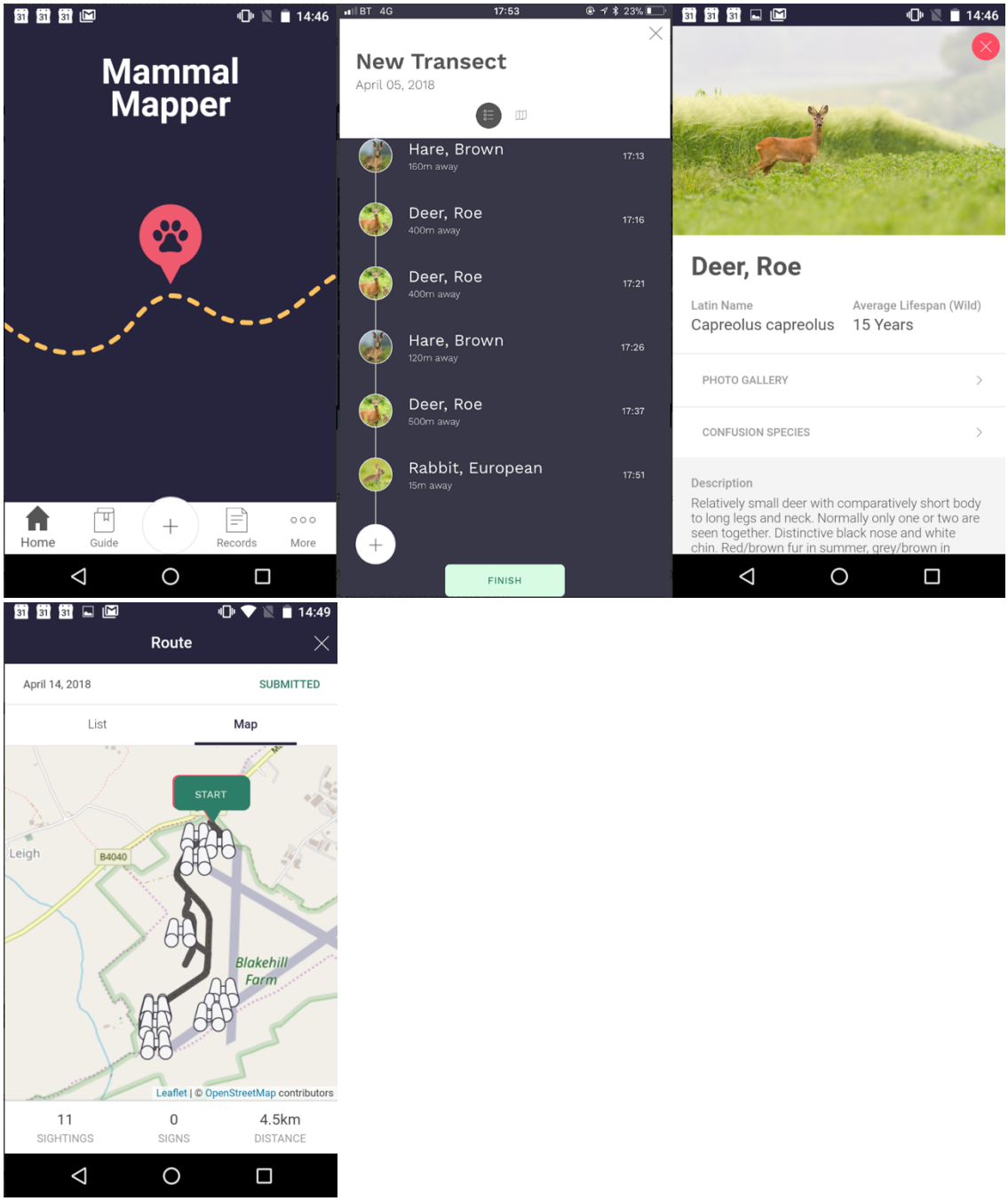

Figure 6 - Screens of the Mammal Mapper application

This application lets the user records mammals during their walks. It groups mammals per specie along with the captures.

Nature Spots

This platform lets the users record and upload user sightings of animals, plants or even fungus around the Leicestershire and Rutland. The user is not obligated to upload a photo, it simply needs to upload information on the sighting to help map it on the distribution map. All records are passed to local and national recording schemes [9].

This platform works more based on a forum, where users show their captures and location.

The following table will compare the applications with the one being detailed on this dissertation. It shows the platforms and the focus of these platforms.

Platform	**Mobile**	**Web**	**Group**	**Capture submission**	**Experts documentation**	**Version**
Vacaloura	No	Yes	Insects (beetles)	Yes	No	Web only
MozzWear	Yes	No	Insects (mosquitos)	Yes	Yes	Android
iRecord	Yes	Yes	All	Yes	Yes	Android and iOS
iNaturalist	Yes	Yes	All	Yes	Yes	Android and iOS
Mammal Mapper	Yes	No	Mammals	Yes	No	Android and iOS
Nature Spots	No	Yes	All	Yes	No	Web only
diversidade.ID	**Yes**	**Yes**	**Insects**	**Yes**	**Yes**	**Android (as of this date)**

Table 1 - Comparison between platforms

The main goal of diversidade.ID is the insects that habit the Baixo Vouga Lagunar ecosystem. None of the applications, besides iNaturalist and iRecord, can accomplish such thing, and even both are more of a global platform than a specific one for specific needs.

Using the capabilities of mobile technology such as GPS, Wi-Fi and Camera, users can capture, document and share their captures to the platform, to be seen by everyone who is invested in this field.

2.2 Review of selected development technologies

2.2.1 Overview of mobile apps implementation strategies

This project will work with two main entities as the gateways of insect's captures – mobile application and web application.

The mobile application works, mostly, for the common smartphone user that wants to help the documenting of any insect species. The web application works in the same way but, instead of using the smartphone to capture, other type of photo capable devices can be used, and its upload done through it.

For that, the mobile and web application needs to have the following main features:

- Capture
- Document
- Captures location
- Captures information
- Species location
- Species information

The web and mobile application have to be synchronized with each other, so they both would need to work and connect with the same back-end services. A capture taken with a smartphone has to appear on the web application, and vice-versa.

The back-end services will serve as the pillar that grants all the synchronization between all platforms, whether it's the mobile or web application. This was the main focus: synchronization. All data has to be synchronized between both applications, all data has to be the same on both applications and any upload must take effect on both applications. By doing the changes on the back-end services, both applications are warned of the changes, and pull any new data.

The web application also needs to be visible for everyone throughout the Internet, and for that a web server needs to be deployed. Deploying our own web server would be costly and not flexible, so using a cloud hosting service is the best way to achieve this. By applying the web application to the remote web server, it becomes visible and accessible to everyone.

The mobile application initial goal was to make it as a fully functioning platform, meaning that, by itself, the platform should be useful. So, with that in mind, the first component to be developed was the mobile application, focusing on the performance and features of this application. Second, the web application was developed – this will work as an extension of the mobile application. Some of the functions from the mobile version will be present on its web counterpart, but the main goal is the development of the mobile application.

2.2.2 Mobile applications development in Android

Android is a mobile operating system created by Google, based on a modified version of the Linux kernel and other software, designed primarily for touch screens, mobile devices such as tablets and phones. There's also an Android version for televisions, called Android TV, for cars, Android Auto, and for watches, Wear OS, each with a custom and specialized user interface.

Android is used in conjunction with a proprietary suite developed by Google, with core services as Gmail or Google Maps being included in every Android version [10]. All applications can be downloaded from the Google Play Store (also included in every

smartphone). All applications are licensed by manufacturers of Android devices certified under standards imposed by Google [11]. Android mobile operating system also as a variant called Android as Open Source Project (ASOP) that is used as a basis for various other platforms that don't fall into the Android ecosystem category such as Amazon's Fire OS, which utilizes its own services compared to Android [12].

Android has been the best-selling OS worldwide on smartphones since 2011 and on tablets since 2013 [13] . As of May 2017, it has over two billion monthly active users, the largest installed base of any operating system, and as of June 2018, the Google Play store features over 3.3 million apps [14].

Applications that run on the Android mobile OS are developed using the Android SDK. They can be developed using Java or, most recently, Kotlin. The Go programming language can also be used to develop applications but with a limited API. Java can be used in conjunction with C/C++ [15].

The development kit includes a big set of developer tools, including debugger, software libraries, a handset emulator based on QEMU, documentation, sample code and tutorials. Initially all Android development was done using Eclipse using the ADT but in December 2014 Google release Android Studio. This platform is based on IntelliJ IDEA and was quickly changed to the standard Android development platform.

Due to the open nature of the Android system, a number of third-party application marketplaces exist for Android, mostly to provide a substitute for devices that are not allowed to ship with Google Play, due to policy violations in certain countries. Example of these third-party stores are the Amazon Appstore or GetJar [16].

2.2.3 Backend services with Firebase

Firebase is a comprehensive mobile platform infrastructure that grants the user the creation of mobile applications with ease, letting the programmer focus on the development of the application itself without the need to worry about the back-end infrastructure. It offers a variety of services such as Cloud Firestore, Cloud Realtime Database, Machine Learning Kit, Cloud Functions, Authentication, Hosting and Cloud Storage. Firebase also offers various analyzing tools that improve applications quality such as Crashalytics, Performance Monitoring and Test Lab.

For the development of this platform the services used were the following: Cloud Realtime Database, Cloud Storage and Authentication.

Firebase Realtime Database

The Firebase Realtime Database lets you build rich, collaborative applications by allowing secure access to the database directly from client-side code. Data is persisted locally, and even while offline, real time events continue to fire, giving the end user a responsive

experience. When the device regains connection, the Realtime Database synchronizes the local data changes with the remote updates that occurred while the client was offline, merging any conflicts automatically. The Realtime Database provides a flexible, expression-based rules language, called Firebase Realtime Database Security Rules, to define how your data should be structured and when data can be read from or written to. When integrated with Firebase Authentication, developers can define who has access to what data, and how they can access it.

The Realtime Database is a NoSQL database and as such has different optimizations and functionality compared to a relational database. The Realtime Database API is designed to only allow operations that can be executed quickly. This enables clients to build a great real time experience that can serve millions of users without compromising on responsiveness. Because of this, it is important to think about how users need to access your data and then structure it accordingly [17].

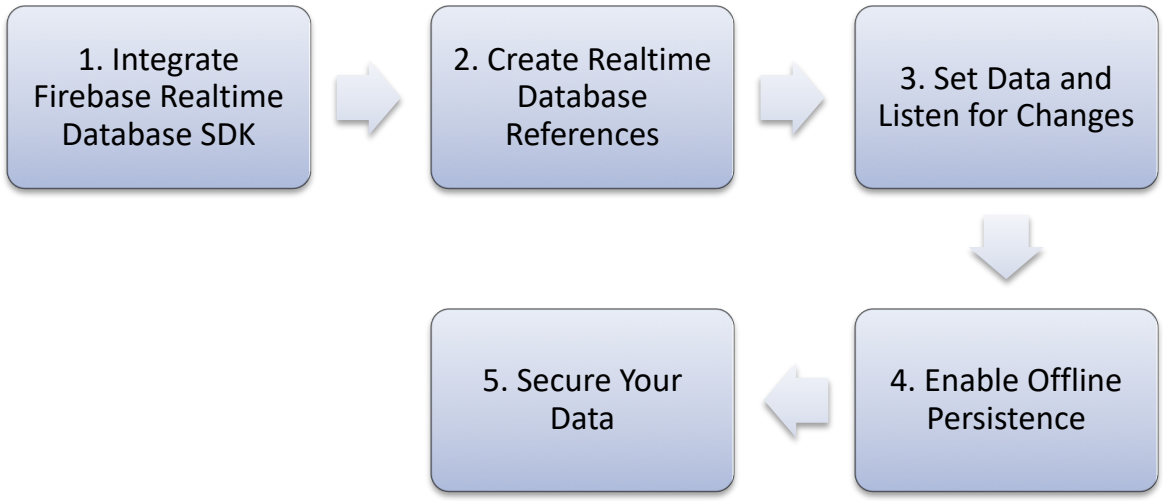

Figure 7 - Firebase Realtime Database Setup

1. Integrate Firebase Realtime Database SDK – Quickly include clients via Gradle, CocoaPods or a script include;
2. Create Realtime Database References – Reference JSON data to set data or subscribe to data changes;
3. Set Data and Listen for Changes – Use previous references to write data or subscribe to data changes;
4. Enable Offline Persistence – Allow data to be written to the device's local disk so it can be available while offline;
5. Secure Data – Use Firebase Realtime Database Security Rules to secure your data.

Every time data changes the client program is warned about it and immediately pulls new data. This is made using a listener.

- Listener: this is part of a programming paradigm called event-driven programming. Event-driven programming is a programming paradigm in which the flow of the program is determined by events such as user actions or messages from other programs or threads. The Realtime Database works exactly like that, every time a change is made in the database it warns the functions that use the listener that something has changed and that a new data pull must be done, refreshing the information shown to the user.

Firebase Storage

Developers use the Firebase SDKs for Cloud Storage to upload and download files directly from clients. If the network connection is poor, the client is able to retry the operation right where it left off, saving users time and bandwidth. Cloud Storage stores files in a Google Cloud Storage bucket, making them accessible through both Firebase and Google Cloud. This allows the flexibility to upload and download files from mobile clients via the Firebase SDKs and do server-side processing such as image filtering or video transcoding using Google Cloud Platform. Cloud Storage scales automatically, meaning that there's no need to migrate to any other provider. The Firebase SDKs for Cloud Storage integrate seamlessly with Firebase Authentication to identify users and provides a declarative security language that lets developers set access controls on individual files or groups of files, so they can make files as public or private as they want [18].

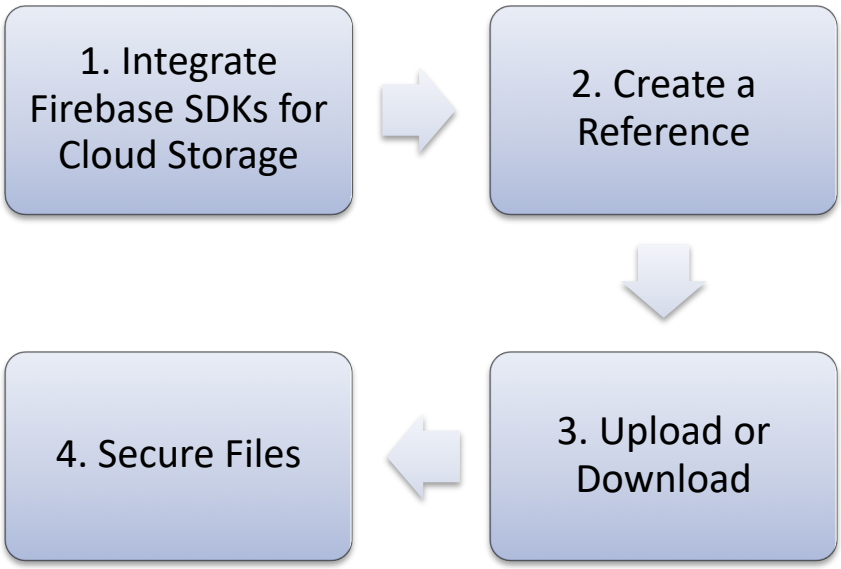

Figure 8 - Firebase Storage Setup

1. Integrate Firebase SDKs for Cloud Storage – quickly include clients via Gradle, CocoaPods or a script include;

2. Create a Reference – Reference a path to a file to upload, download or delete it;
3. Upload or Download – Upload or download to native types in memory or disk;
4. Secure Files – Use Firebase Security Rules for Cloud Storage to secure your files.

Firebase Authentication

To sign a user into the app, first get authentication credentials from the user. These credentials can be the user's email address and password, or an OAuth token from a federated identity provider. Then, pass these credentials to the Firebase Authentication SDK. Firebase backend services will then verify those credentials and return a response to the client. After a successful sign in, access to the user's basic profile information is given, and the developer can control the user's access to data stored in other Firebase products. He can also use the provided authentication token to verify the identity of users in his own backend services. Firebase Authentication offers two ways of signing/logging in a user with FirebaseUI Auth and Firebase SDK Authentication.

2.2.4 Content management with WordPress

WordPress is a free and open-source CMS based on PHP and MySQL. It's the most popular blogging system [19] and it offers two big features: templates and plugins. As implicitly stated before WordPress is usually associated with blogging but can be used to design and implement an array of types of websites. It's estimated that it's used around more than 60 million websites [20] including 30.6% of the top 10 million websites as of April 2018 [21]. WordPress has to be installed on a web server, either on a hosting service or an own computer running the software package Wordpress.org in order to serve as a network host in its own right.

WordPress has a web template system using a template processor. Its architecture is a front controller, routing all requests for non-static URIs to a single PHP file which parses the URI identifies the target page [22].

2.2.4.1 Themes

Users may install and switch between different template themes. Themes allow the user to change the look and functionality of a WordPress webpage without changing the core code or anything pre-built with WordPress. Every website needs at least one theme to be applied – each theme is made using PHP, HTML and CSS. Themes can be installed using the included theme store or by uploading via FTP. All PHP, HTML and CSS can be changed by the user, creating a child template that can be freely modified to meet the user's needs. JavaScript can also be implemented to run alongside PHP code.

2.2.4.2 Plugins

WordPress allows the installation and use of custom plugins, expanding the platform's capabilities or simply simplifying some development processes. Such plugin is CSS Editor, used by this platform, to insert custom CSS per page instead of using the style.css file to implement all custom changes.

3 Use cases for participatory entomology

3.1 Activities and processes in citizen science

On this platform there will be two relevant users: the citizen user and the collaborator user. The first one is the basic user, the one that makes captures and can see information, and the second one is the platform's power user. The collaborator user can edit information and document species.

Both the collaborator user and the citizen user have one common responsibility: make new captures. This is the point where the responsibilities of both users differ – one simply captures and waits for approval, while the other takes a photo and can immediately document the species. The latter can also document citizens captures.

What next follows is the description of a simple workflow on this project: there's two users, John and Angie. John is a normal user, someone with no knowledge of entomology but is using the application to help with the project; Angie is a power user that is specialized in entomology. Both create their accounts, but John doesn't check the mark for Collaborator, while Angie creates her account and checks the Collaborator mark. With both accounts created, John decides to take a photo of an insect he just found. The moment the photo is uploaded to the database, Angie will be warned through a notification of her smartphone. By checking the notification, she can see that new captures were made and are waiting for approval. Clicking on the notification, Angie is redirected to the "Pending" activity, the activity then shows all the pending captures waiting for approval. Angie clicks on the picture and is redirected to the capture information – she can check the location to help identify if it belongs to an existing specie or, if it doesn't, can be documented as new specie. Angie clicks on the edit details button and is redirected to the page of edition, where she can reject immediately fill in the information and accept the capture. When the capture is accepted, John will immediately see that his capture was documented and already belongs to a specie. He can check the details of the specie he just captured and other captures of the same species.

3.2 Use cases description

There are a variety of use cases that should be defined for the project, with every one of them being a feature of the project itself. These use cases also dictate how the system will be structured. This chapter will go through all identified use cases during the early prototyping of the system.

There are two use cases scenarios: the web platform and the mobile platform. The web platform will have an extra actor that will work as an administrator. This administrator will have different functions from the collaborator user, even though to be an administrator, the user needs to be someone that knows about the field of study, such as an entomologist or collaborator user. The rest of the use cases are the same for both platforms, as they will be designed as an extension of each other.

Consider de following images:

Figure 9 - Mobile application use cases

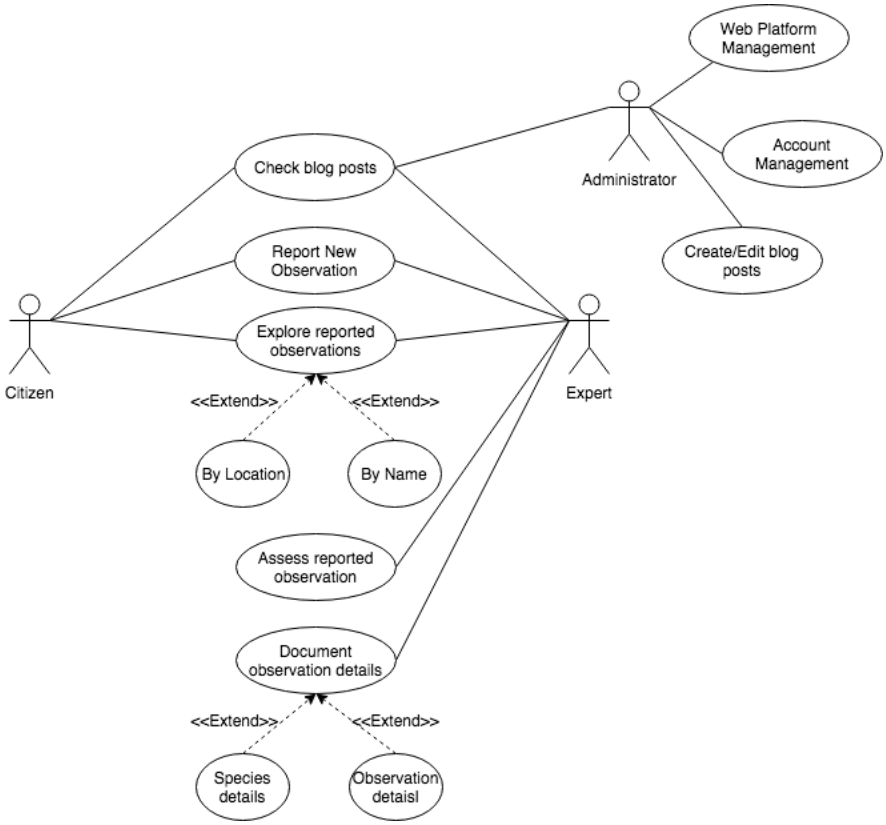

Figure 10 - Web application use cases

As already stated, in the second figure (Figure 10), the administrator is a different entity from the collaborator user. Both shouldn't be mixed as they don't have the same goals nor the same actions.

3.2.1 Use cases description

3.2.1.1 Mobile

Use Case	Primary Actor	Brief	Basic Flow
Report New Observation	Citizen/Collaborator	The user reports a new observation	The user logs into the application, chooses an activity from the initial screen and clicks on the camera button on the down left corner
Explore Reported Observations	Citizen/Collaborator	The user explores other captures	The user logs into the application, chooses the

			Sighting Gallery activity
Assess Reported Observations	Collaborator	The collaborator user recognizes there are pending captures waiting for documentation	The user logs into the application, clicks on the notification received regarding pending captures
Document Observations Details	Collaborator	The collaborator user documents details regarding a pending capture	The user logs into the application, clicks on the notification or goes to the pending captures activity, selects one capture and clicks on the edit details button

Table 2 - Mobile use cases table

Consider the first image (Figure 9): this describes the use cases for the mobile platform. There are use cases common to both actors of the platform such as Report New Observation or Explore Reported Observations. There are two extra cases that extend the functionality of the Explore Reported Observations case, one that lets the user search based on location by using the map activity, and one that lets the user search based on name, whether it's the scientific name or the vulgar name.

There are other two use cases that are exclusive to the collaborator actor, Assess Reported Observation and Document Observation Details. The first one describes how the collaborator user can assess any new capture made by another user, recognizing it as a valid capture. The second one describes how the collaborator user can document/edit details regarding a specie as a whole, changing all information of all the captures belonging to it, or change the details of a specific capture without affecting the rest of the captures of that specie.

3.2.1.2 Web

Use Case	Primary Actor	Brief	Basic Flow
Web Platform Management	Administrator	The administrator can manage all the accounts conneceted to the website	The administrator goes to the CMS user interface and can change the design and source code of the web application
Account Management	Administrator	The administrator can manage the	The administrator goes to the CMS user interface and

		accounts in the database	manages the accounts
Create/Edit Blog Posts	Administrator	The administrator can create new blog posts for other users to see	The administrator goes to the CMS user interface and creates a new entry for the blog
Check Blog Posts	All	The user can check blog posts made by the administrator	The user goes to the blog page and checks blog posts
Report New Observation	Citizen/Collaborator	The user reports a new observation	The user logs into the application, chooses an activity from the initial screen and clicks on the camera button on the down left corner
Explore Reported Observations	Citizen/Collaborator	The user explores other captures	The user logs into the application, chooses the Sighting Gallery activity
Assess Reported Observations	Collaborator	The collaborator user recognizes there are pending captures waiting for documentation	The user logs into the application, clicks on the notification received regarding pending captures
Document Observation Details	Collaborator	The collaborator user documents details regarding a pending capture	The user logs into the application, clicks on the notification or goes to the pending captures activity, selects one capture and clicks on the edit details button

Table 3 - Web application use cases table

The mobile and web platform have most of the use cases in common, with both actors playing a role in this platform as well. All the use cases are the same and they describe the same features to be implemented.

Besides extra use cases, this platform also has a new actor. This new actor is the administrator of the system. The administrator role will let the actor manage the web platform, from design to features, account management and create or edit blog posts. These blog posts are also visible to both the citizen and collaborator actors.

The collaborator user can also be an Administrator, since the administrators of the platform need to know or have some kind of background in entomology to be able to manage the web platform.

4 Proposed architecture and methods

4.1 System modules

For the solution to work, a variety of modules have to work with each other (Figure 11). There are certain modules that are obligatory for the platform to work, such as databases and applications.

Since the platform main goal is to document species, it needs to have a database to save all the information. The information also needs to be constantly synchronized between the database and the platforms.

For the captures to be made and uploaded, a mobile application needs to be deployed. Some captures can be done without a smartphone, so to accommodate that feature, a web application also needs to be deployed, giving the user to upload high quality images from outside the mobile application.

The web application needs to work as a blog as well and be deployed in a web server to be accessible to everyone. This application will need a database to store every information regarding the content of that web application and needs a cloud solution to be deployed so it becomes accessible throughout the internet.

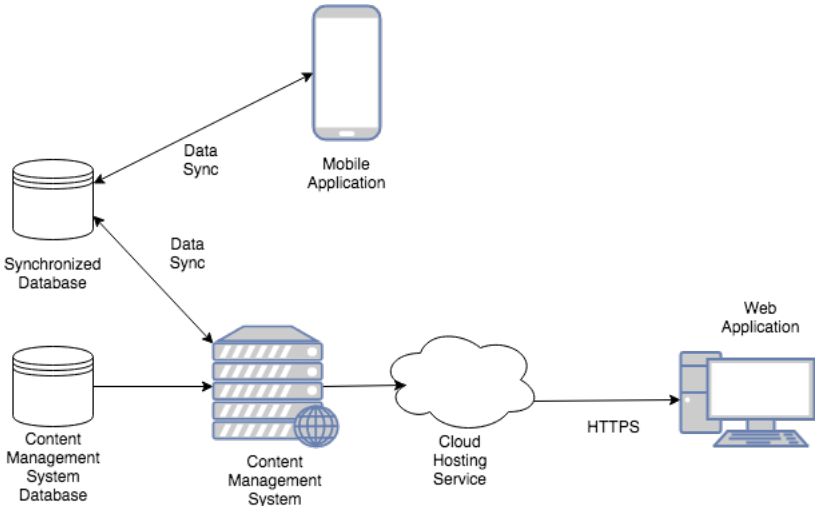

Figure 11 - System module architecture

4.1.1 Key Modules

The database, an always online, synchronized and organized database needs to be deployed, with all other components connecting to it. That database contains all species info, such as description, vulgar name and ecology, and all its captures. All that information needs to be instantly accessible by the mobile and web applications.

The mobile application will need to connect directly to the database, fetching and uploading any new data that's inserted into it. And all data will need to be visible on the mobile platform, with the minimal delay as possible, in a way to make the user experience enjoyable.

The web application will be designed as an extension of the mobile experience. Not only users will do the same as they do on the mobile platform, but they will also be able to do more. The extra feature will be a blog, managed by the web platform administration, where users could see news, videos or images posted by an administrator. This application will need to also be visible for everyone – initially, an own server would be deployed, but thanks to costs and time, another approach will need to be made. By using a cloud hosting service, the web application can be up and running, without the need to pay attention to management or costs.

Inside the web application, some users have to be designated as administrators. To not mix platform collaborators and administrators (as they aren't the same), a new database will be used to save administrator credentials, and everything related to the web application. Remember, the main database will be only used for captures and species documentation, citizen and power users, also called as collaborators. The web application database will only contain data related to the web application itself and not the overall platform.

4.2 Modules interactions

The base principle is that both applications, mobile and web applications, must have the same back-end service pushing data to them. A cloud solution must also be applied for the web application to be accessible to anyone, and a content management system will need to be employed so the web application can show a blog for any information sharing.

When a user creates a new account, that account should be registered in two places – authentication system and in the database. The authentication system will handle all the sign-in and login operations, as well as all the management from the accounts. This will be a separate system from the database, as the database will only be used as storage and privileges verification. Inside the database, all users will have their own entry, where all the captures made will be saved.

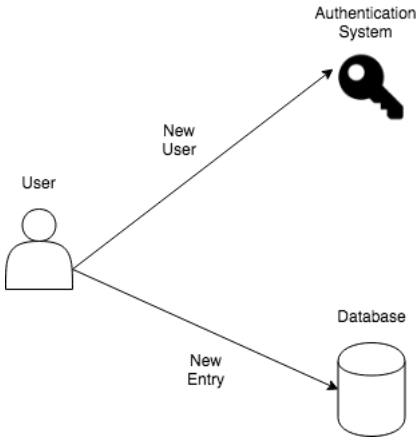

Figure 12 - User registration

When a user decides to take a capture, that capture is sent to the database itself. It will create a new entry for that capture under the user's branch (Figure 12). After the upload is done, the new information needs to be pushed to all the devices connected, whether it's a mobile application or the web application. Since this capture still isn't documented, the collaborator users should receive a notification warning them that a new capture has been made and needs to be documented. Only the collaborator users can see the pending captures, and the owner of the capture can see his capture through his capture's gallery.

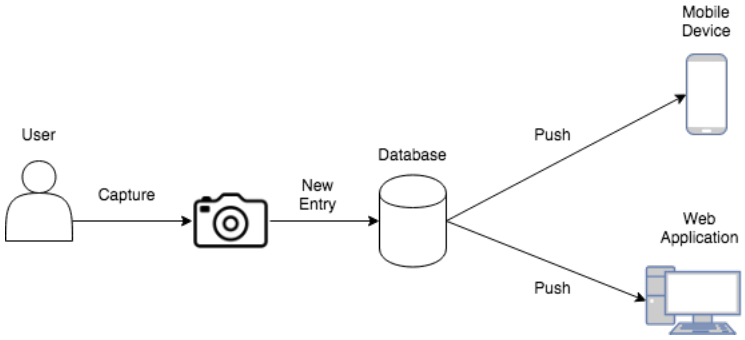

Figure 13 - New capture workflow

After doing the documentation of the capture, the collaborator user uploads the new information of that capture. That information is sent to the database and it pushes to every device the new data that has been added (Figure 13).

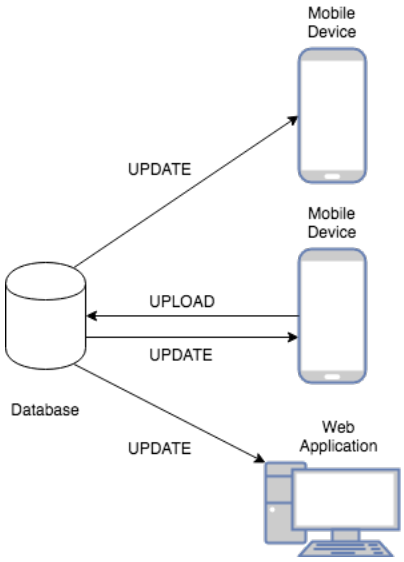

Figure 14 - Documentation workflow

All data is immediately synchronized between devices after any change is made to the database. Both web and mobile application will show the changes, since both of them read from the same source of information (Figure 14).

Now, both collaborator and citizen user can check the information regarding that capture and all other captures.

4.3 Managed objects/data structures

All the information saved on the main database will be saved as a structured object, making it easy to access and understand. It also will help with the algorithm navigation if the database is well structured.

There are four fields that should be common to all species: species name, species vulgar name, ecology and description. These are the fields that should distinguish species from each other, so all captures belonging to the same specie should have these fields in common (with the same value). So, instead of adding that information to each capture, the best way to approach this is to make them species attributes. Consider the following image:

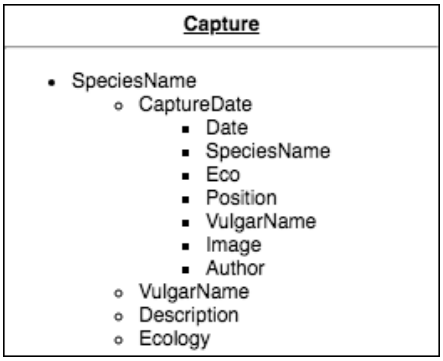

Figure 15 - Capture object class structure

As seen in the image above (Figure 15), four fields are common to a specie – species name, species vulgar name, ecology and description. The species name should be used to distinguish species from each other, being used as the parent node for a specie in our database. That way, it will be easier to cycle through all species. The species vulgar name is another field that is common to a specie, but since the species name should be used as the differentiator between species, the vulgar name should be considered an extra field inside the node itself. The description will contain the species description such as, behaviour, particularities from that specie. The ecology field will contain the conditions of how the species live.

There are a variety of use cases that modulate how the database should be developed. Since the mobile application will work as a feed, the way it gets its information from the database needs to be strictly defined to prevent any errors.

4.4 User experience

The mobile application is one of two ways of interacting with the platform diversidade.ID. This can be used by everyone, whether they are citizen users or collaborators. The mobile application was designed to be the main focus of interaction with the platform.

The main goal of this application is to be used by an ordinary user. Considering that not every user has basic technological knowledge, the application was developed with simplicity in mind. Big and visible icons that self-describe and inspiration from other big applications, gave the possibility to design a simple yet powerful application that can help entomology.

First version prototype

First, the prototype designed was made to show a first glimpse of what the application could do, showing how it could look like and what would it do. Consider the following image:

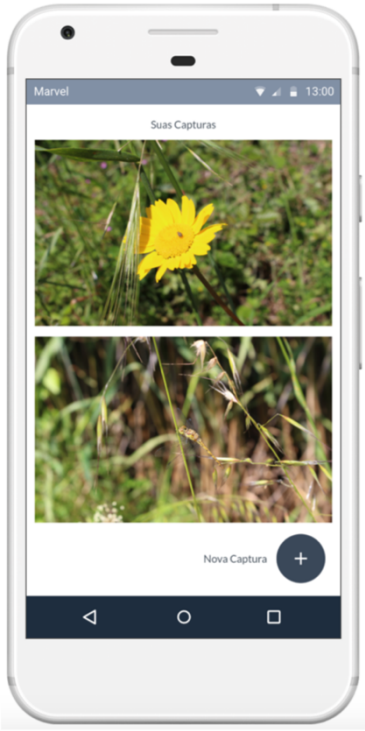

Figure 16 - Main feed mock up

This image (Figure 16) shows the initial vision for the feed. It's noticeable the Instagram inspired look, showing minimal text and only the image. It's an infinite scroll that shows the user's pictures and, on another activity, other user's pictures.

The "New Capture" button is clearly visible, informing the user that a capture can be done by simply clicking the button. It's a clear and simple design and motivates the user to use the application by seeing what captures the other users did.

When a user takes a capture, it's uploaded to the database and waits for approval and documentation from a collaborator. After the approval, the image can finally be seen, and all its information can be checked out by another users. This is the Capture Information Activity:

Figure 17 - Capture information mock up

This is the prototype made for this activity (Figure 17). It shows the image selected and all the information added by the collaborator, including the location where the capture took place. There's also a button in the activity, only visible to the collaborators, that lets them edit the information.

Since a high number of species will be documented, a list containing all species as a way to gather all information about it is a good solution. Each entry of that list will work as a hub for the species, showing all information about it, where the captures were made and all captures from that specie.

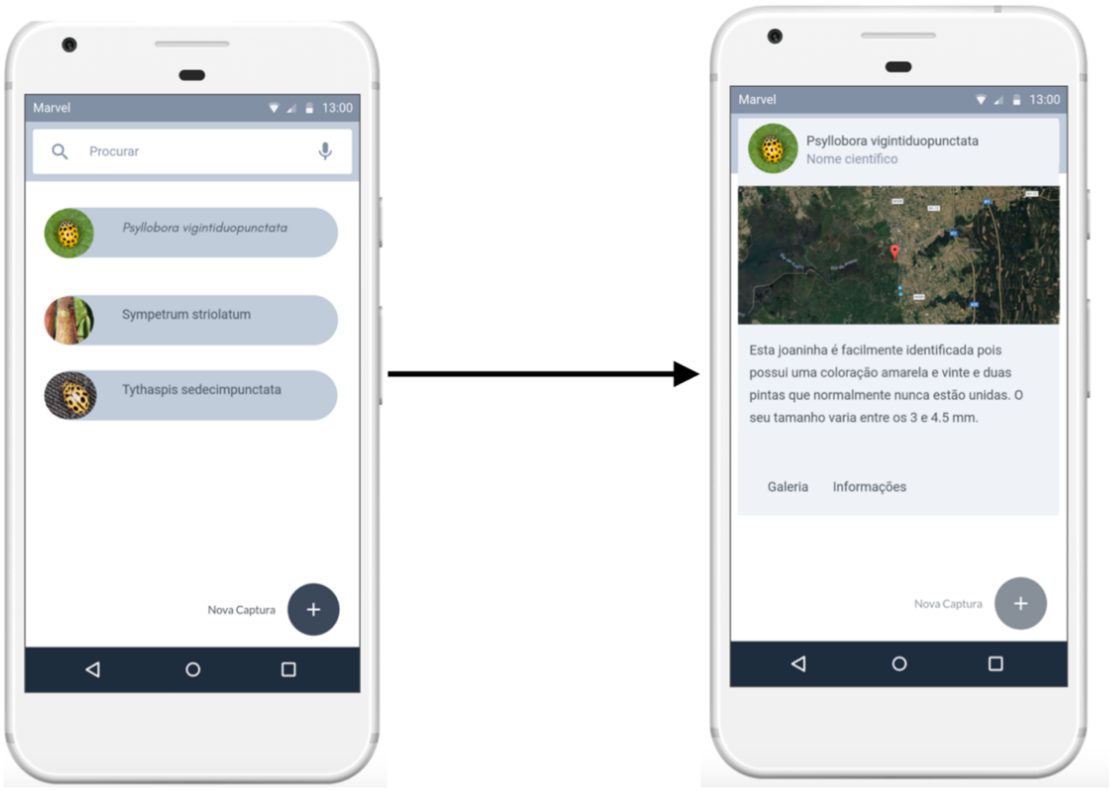

Figure 18 - Species guide mock up

The initial thought was to show the list of species (Figure 18), click in one and open an extension showing two options: Gallery or more information. If the user clicked on the "Information" button it would lead him/her to the following activity:

Figure 19 - Species information mock up

On the top there's a visible tab section that would divide all the information into: Information, Gallery and Map (Figure 19).

When documenting a new capture or editing an already documented one, the collaborator user will have its own activity to do such. For this activity there wasn't a prototype made for it, but it should follow the same design language used throughout the previous mock ups.

Capture location is a big feature of this platform, as it shows where the capture took place and can help recognize species gatherings. For this feature, a simple Map fragment should be used – it should show the exact place where the capture was taken.

There's also a Species Map that should show species location on the map. By clicking on the markers, the users need to be redirected to the species activity, showing all the information and all the captures taken.

5 Implementation

5.1 System deployment and implementation technologies

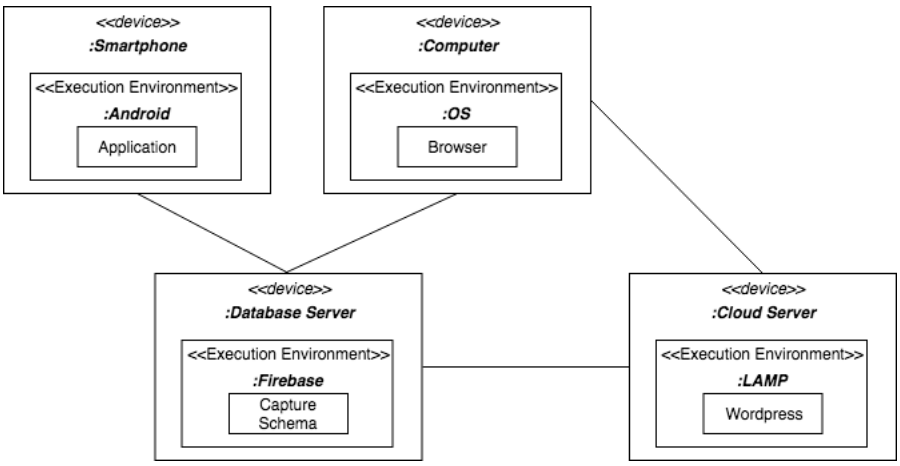

Figure 20 - System deployment diagram

The deployment diagram above (Figure 20) describes how the deployment of the system was made. There's a mobile application and a web application that will work together through the Firebase Realtime Database. They both will read from the same database, keeping the information synchronized between them.

The web application will be accessible through a local computer, by just typing the address on the browser. The web application is stored in a remote web server, not managed by us as it is a cloud web server. This cloud server will also have connection to the Firebase Realtime Database, where it will fetch/push any data from/to it.

The following shows a list of the technologies used and their role in the system:

- **Back-End:**
  - **Storage:**
    - **Firebase Realtime Database**
    - **Firebase Storage**
    - **MySQL**
  - **Web Hosting:**
    - **Digital Ocean**
  - **Runtime Environment:**
    - **/LAMP**
  - **Authentication:**
    - **Firebase Authentication**

- **Front-End:**
  - **Mobile Application:**
    - **Android**
  - **Web Application:**
    - **WordPress**

5.2 Mobile app implementation

The mobile application is the main way of regular users to make captures. Everyone has a smartphone, so the application has to be feature-rich to make sure that it's capable of acting as a standalone platform.

5.2.1 Architecture

For the mobile application to work it needs 4 main components:

- Firebase Realtime Database
- Firebase Authentication
- Firebase Storage
- Android Device

5.2.1.1 Firebase Realtime Database

All data is stored as JSON Objects in the Firebase Realtime Database. Every time the Android applications needs information, instead of always fetching from the database, it reads from what it already has and displays it. It only fetches data from the database when the application starts or when new data is inserted into it. Consider the following image:

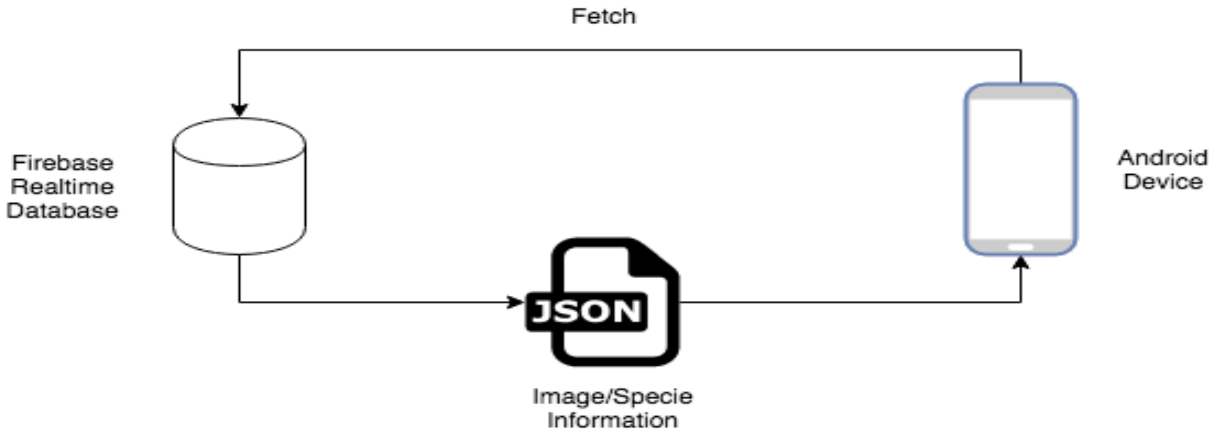

Figure 21 - Firebase information exchange

Using Firebase own methods, it puts a listener on the database and, every time the data changes, all listeners are activated and the data on the mobile device is updated (Figure 21). For example, the "Species Guide" activity shows a list containing all the species. This

activity has a listener on the Species branch in the Firebase Realtime Database. Every time a new specie is added to this branch, the listener regarding this activity will trigger, updating the list of species it has on the device. That way, the data is never out of date.

If the mobile device goes offline, the application will still read the local data it already has. It's an already out of date data but it prevents the application from not showing anything at all. Attention, this only shows the information from the activity where the user is at when the connection goes down. If the user tries to go to another activity, it will simply show nothing. For example, if the user is on the "My Sightings" activity, then it can see the captures information and position. This works because, every time the application gets information from the database, it gets as a JSON object and all its information is shared between consequent activities. This prevents the application from trying to fetch individual information regarding a capture from the database – and if the connection goes down, that information can still be seen.

5.2.1.2 Firebase Authentication

Firebase Authentication is the login method used for this platform. The method used for the login is through email and password, the most common used method of registering. Firebase Authentication also deals with the redefinition of passwords, email verification and management of accounts through the Firebase console.

The first time a user registers, its credentials are stored in the Firebase Authentication platform. Depending on whether the user checked the mark "Collaborator" or not, the Firebase Realtime Database will create a new entry in the "Accounts" branch with the child "isPro" set as true – this means that this user is a collaborator (Figure 22). This branch is designed, mostly, for all the collaborator users and is used by the applications to check if the user has the privileges of a collaborator. Those privileges include editing information, checking pending captures, etc.

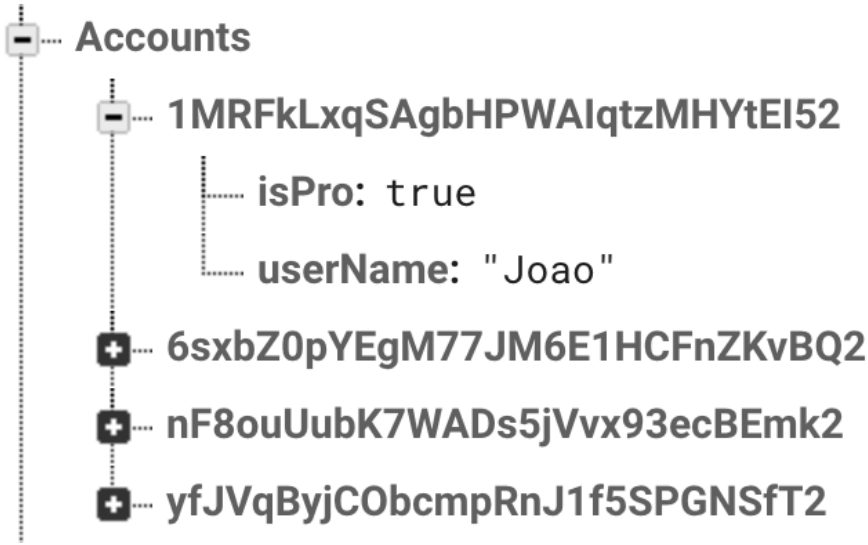

Figure 22 - Accounts branch in Firebase Realtime Database

For example, every time the user gets into the “Species Information” or “Capture Information” activities, the algorithm goes through the “Accounts” branch and checks if the current user logged in has the Collaborator permissions. If it does, the activity will show the option to edit the details of the specie or capture.

To make the login process easier, and to avoid the need for login every time a user opens the application, the login method has a listener that knows if the user is already logged in on the smartphone. In case the user is already logged in, it goes automatically to the application’s initial screen.

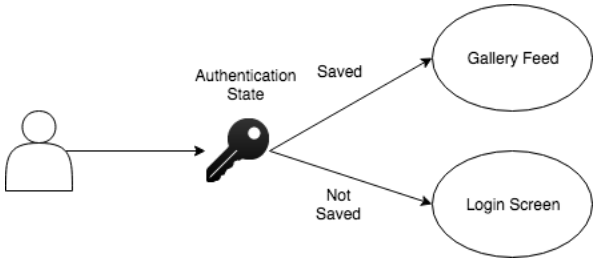

Figure 23 - Login workflow

As seen in the image (Figure 23), if the Firebase has the Authentication State saved, the user is automatically logged in. If not, then the user must fill its credentials.

5.2.1.3 Firebase Storage

The Firebase Storage and Realtime Database work together for image storage. Since the Firebase Realtime Database doesn’t support images, the way to associate the image to the correct capture information was to create a field on the capture object called “URL” (Figure 24). That field will contain the link to the correct image saved in the Firebase Storage.

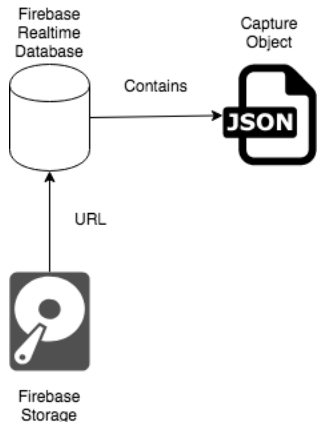

Figure 24 - FIrebase storage workflow

To form the complete capture object with all the fields, the Firebase Realtime Database needs the Firebase Storage to share the URL of the capture taken.

5.2.1.4 Android Device

The objective of the mobile application is to be able to hold on its own as a platform, letting both type of users interact with the platform solely by using it. Whether it's a citizen user or a collaborator user, the application needs to allow the user to capture, view and document captures.

Data Usage

First, an explanation of how the data flows through the application. Since this is an application where the images take the central spot for the user, all those images have to be read from somewhere. In fact, they are read from a branch every time the user goes to a gallery activity. The whole image is not actually read, just a small one, like a placeholder, in order not to waste too many resources such as memory or mobile data. The application not only reads the image itself, but it also reads all the other information regarding that capture. In fact, it reads the data as a capture object, containing all the information regarding to it.

This approach lets the application read data once and assign it to the image correctly. So, by reading this data once, it stores it on the application and consequent activities that use that data can read without the need to connect to the database again. This saves data and memory, and, in case the smartphone goes offline, still lets the user check information regarding a capture. This approach doesn't work as a solution for all the activities in the application– if the user, for example, goes to another activity that is not connected to the current one, the data fails to read, and no information is shown. For example, if John is checking his current captures and suddenly the device goes offline, he can still check the information regarding his captures. But if John goes to another gallery activity like "Gallery", because it's not connected to the previous one, it needs to fetch the data again. No internet connection is detected, so no data is shown to John.

Such approach can consume much memory, but it doesn't consume much data. The memory consumption can be kept at a minimum by using dedicated software or by using algorithms designed for this specific problem.

Image Loading with Glide

As previous said, one of the biggest problems while designing the mobile application was memory management. Being a heavy image-based application, with some images being of really high resolution, managing the memory is one of the important aspects to take into consideration. At first, with a low number of photos, the application has no problem rendering all images, with the memory footprint being quite low. The more images are loaded into the database, the more images the application needs to download and show to the user. The images are always stored in cache, to prevent the user to download the images every time it wants to access it. But since this cache is a limited memory pool that needs to be shared with other applications, storing all info from the application on this memory cache would lead to the Android system to start closing applications, resulting in crashes or other problems. It's easy to forget the images being loaded into the cache, leading to a memory leak. This would be even easier to achieve on low-end phones running Android KitKat 4.0, as those devices have

low amounts of memory reserved for cache. So, to prevent this, treating the image loading and activity management is paramount for the application's performance.

To deal with the image loading and caching, a third-party library called GLIDE was used. GLIDE supports fetching, decoding and displaying of images. The primary focus of this library is to make lists of images as smooth and fast as possible. It also facilitates the loading of remote images with a HTTPUrlConnection based stack. By using GLIDE, the memory footprint was cut in over half of what it would be if this third-party library wasn't used.

This software is based on another third-party library called Disk LRU Cache. What this cache does is limit the number of bytes that are stored on the filesystem. When the number of bytes exceeds the store limit, this cache will remove entries in the background until the value is below the limit again. This limit imposed by the cache is not strict – if the application needs more cache memory while the cache is still deleting old entries, then the cache will exceed that limit value, temporarily.

GLIDE works based on this principle but works in a different way, as it reuses bitmaps. Every time GLIDE needs to download a new image, instead of calling the Garbage Collector to clear the old entries, it reuses the memory already allocated for a previous image to load the new one. This way, it grants the reuse of memory, with a big advantage of not calling the Garbage Collector, that usually slows down the performance.

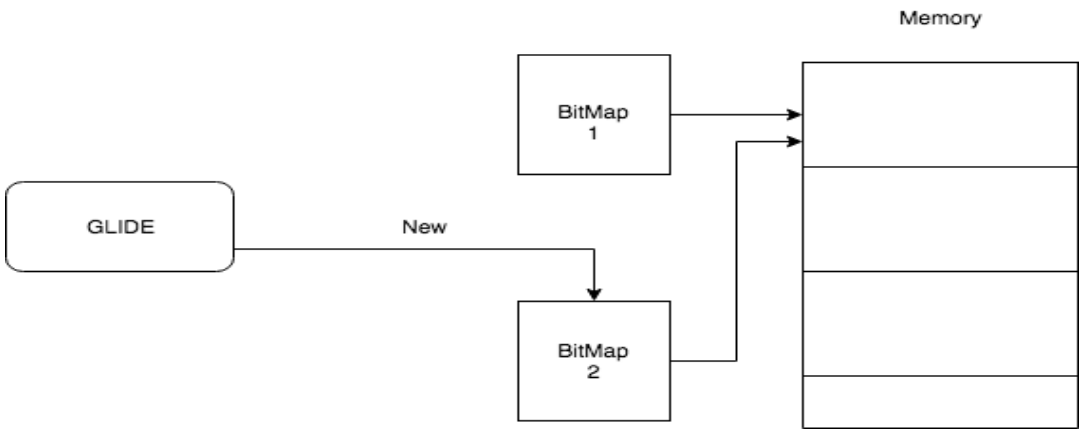

Figure 25 - GLIDE memory management

Consider the BitMap 1 to be the older and the BitMap 2 to be the new one fetched by GLIDE (Figure 25). The memory being used by BitMap 1 will be reused for the BitMap 2 call. That way, memory management becomes less problematic and the chance of getting an "Out of Memory" error gets almost non-existent.

GLIDE will, through an already existing method, load the image through a URL given by the Firebase Realtime Database. As said before, that URL is a direct connection to the image on the Firebase Storage. Consider the following image:

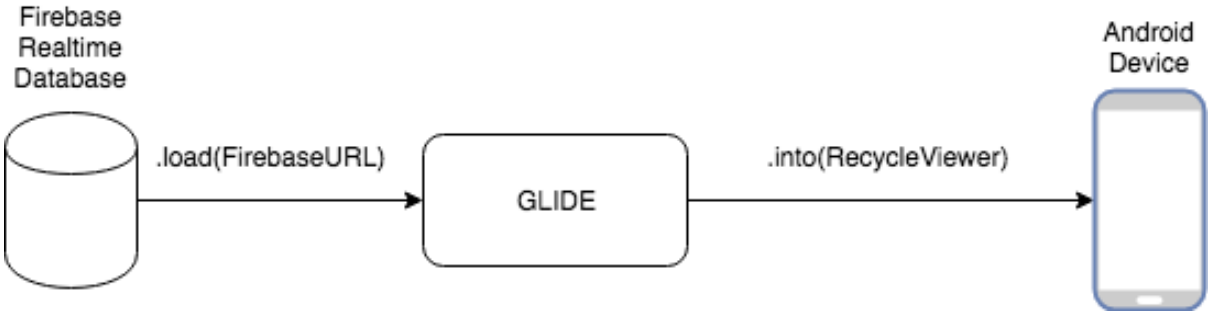

Figure 26 - Image attachment with GLIDE

As the image shows (Figure 26), the GLIDE software uses an own method called “load” that, by giving it the image URL from Firebase as the argument, fetches and downloads the image, putting it on cache.

Custom Views

After doing the fetch, the image needs to be put somewhere – on this project a custom RecyclerViewer was used. A RecyclerViewer is a structure made to display a list of elements based on large data sets, that can change frequently. This structure is made with holders, objects that hold individual information regarding another object - for example, a holder for this project will display the image, the name, the author, a button for the capture’s location and a button for more information.

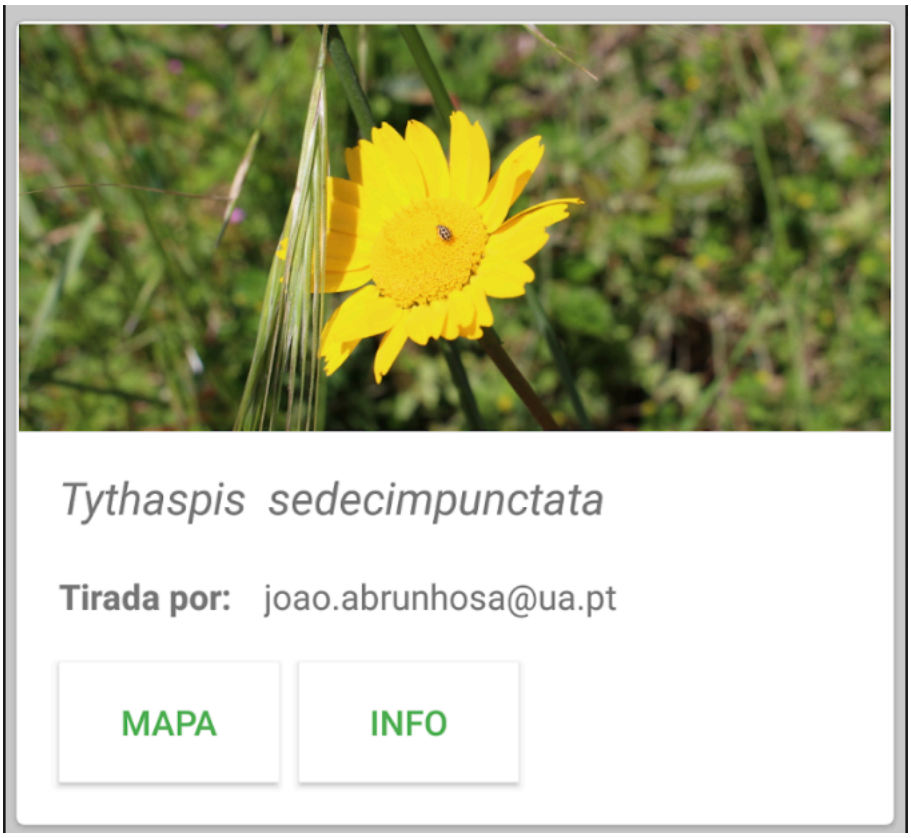

Figure 27 - Custom RecyclerView holder

This holder (Figure 27) will be the object where the GLIDE software will put the image that it's holding on cache. When not using any RecyclerViewers, GLIDE will load the image directly into the ImageView of the activity.

As said, the approach used for this platform was to read all data and attach it. These holders are where that data is attached. For each image, its data is attached to the correspondent holder.

Capture Location with Google Maps

Google Maps is a web-mapping service developed by Google. This service offers street maps, satellite view, 360º panoramic views of streets, real-time traffic conditions, and route planning for travelling by foot, car, bicycle and public transportation.

On this application, Google Maps was use for map location of the captures. The moment the user makes a new capture, its location is automatically calculated through GPS or Internet Provider and used on Google Maps to show its location.

The location is stored on the capture object using the coordinates captured of the time of the upload (Figure 28).

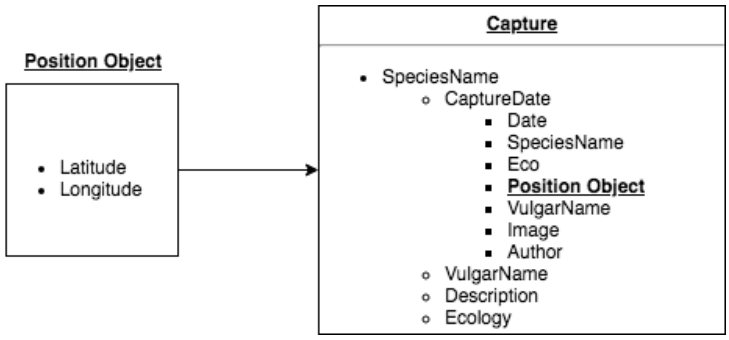

Figure 28 - Position placement in the Capture object

5.2.2 Implemented features

The application needs to work as an image feed, with the capabilities of information edition. As we have two different type of users, the application won't offer the same features for both type of users. Consider the following lists:

- Citizen:
  - New Capture
  - Check Species Information
  - Check Capture Information
  - Check Species Location
  - Check Capture Location
  - Check Other User's captures

- Collaborator:
    - New Capture
    - Check Species Information
    - Check Capture Information
    - Check Other User's Captures
    - Edit Capture's Information
    - Edit Species Information
    - Check Species Location
    - Check Capture Location
    - Approve/Reject User's Captures

As seen on the list above, there are clear differences between the two users: one can be considered more of a power user, while the other has the basic functionalities of a citizen user. This is the principle on how the mobile application implementation was done. The following features will follow the same John (citizen) and Angie (collaborator) story used until now, for coherence.

New Capture

After doing the login, John decides to take a picture of an insect he just found. He can do it from every gallery activity inside the application. When John clicks the button, the Camera API informs of the intent to take a picture. The application suspends until a picture is taken and returned as a result. If the result is different than null, the data received is accepted, and both its time stamp and path are used for the upload of the picture. If the user has the location enabled on its smartphone, then the upload will proceed. Because the upload is made inside the same method of the location capture one, if the user doesn't have smartphone location enabled, no upload will happen.

When the upload begins, it shows the user feedback of how the upload is evolving, through a progress bar on the top of the feed. After the upload is done, the Firebase Realtime Database stores the capture on the branch for pending captures and John needs to wait for approval of his capture (Figure 29).

Figure 29 - Branch for pending captures

At this point the Angie has been warned, through notifications, that a new capture is waiting to be documented (Figure 30).

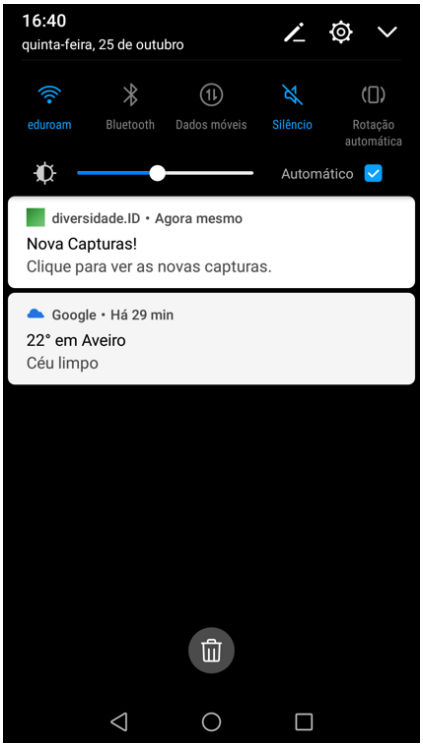

Figure 30 - Notification service

Edit Capture's Information

After John's capture has been uploaded, Angie now has the option to check that capture. As explained before, Angie receives a notification that a capture is waiting for approval. By clicking on the notification, Angie is redirected to the pending captures activity,

where she can choose John's capture and edit its details. Since Angie is a collaborator user, she can see the option to edit the capture. Clicking on the edit details button, Angie is redirected to the capture details activity where she can document that capture.

After getting his capture documented, John can now see, through his capture's activity, that his capture now has information. If his capture was rejected, John would see that his capture wouldn't be on his profile anymore.

This also works after a capture was already documented. Let's imagine that Angie wrongly documented the capture and noticed it. She can go to that capture's activity, go to the details activity and edit the fields. Doing that would change that capture information.

This works by changing the data on the capture object itself. Since the structure of the data is prepared for this type of operations, by having these fields as part of the object itself, it's easier and more effortless for the collaborator to apply these changes.

In the case of the capture changing species completely, the algorithm removes the entry of that capture from the wrong specie and moves it to the new one.

Check Capture Information

After John has his capture documented, he can now check the information added. Clicking on his capture, John is redirected to that capture's information activity. There he can see the specie name, vulgar name, who took it and its location.

This information can only be edited by an collaborator user such as Angie. John has no control over it.

The information regarding a capture is done during the gallery activity prior to this one. By doing that, if the user goes offline, the information regarding that capture can still be seen. Since this information is readily available at the call of the capture information activity, the algorithm only needs to assign the information to the respective element.

Check Species Information

By clicking on the specie's name, John can be redirected to the Specie's Hub activity where he can check all information regarding that specie, including other user's captures, their locations, who took it, description, ecology and vulgar name.

Other user's capture will be gathered in a feed at the bottom of the activity. This reads all the captures saved in the Firebase Realtime Database from that specie.

Since the species information is much more detailed than the capture's ones, the information isn't fetched on the guide activity, in similarity as the capture information one. In this case, when the user selects a specie to check, the algorithm passes the name of the specie to the new activity and uses it to fetch information regarding that specie directly from

the Firebase Realtime Database. After fetching the information and all the captures related to it, it maps the elements to their correct location.

Check Species Location

This activity will let the users use the map to search for species. By using the search function above the map, it will lead them to the desired specie, with the option to be redirected to the specie's information page.

This will help the users identify the location of the species throughout the Baixo Vouga Lagunar.

The specie location is fetched when the activity also fetches the rest of the information. Since the species information activity fetches all captures related to it, it also fetches the location of those captures. Clicking on the Map location button, sends those coordinates as Position Objects to the map activity and displays them.

Check Capture Location

When navigating through a capture's information page, the user has the option to check where that capture was taken. The position seen on the screen belongs to the capture itself.

The capture location is fetched on the activity that precedes the capture information activity. The coordinates are passed to the map activity as a Position object and is correctly displayed on it.

Check Other User's Captures

This is a feed that will show to any user the captures made of all other users. It's a feed that shows every photo taken and it was one of the main problems of this application. This will be discussed on the next chapter.

This feed works exactly like the current user's image feed but, instead of only getting the user's captures from the User branch in the database, it cycles through all the sub-branches on the Species branch in the database. Doing that shows all captures from all species, made by a variety of users.

Edit Species Information

This is another feature only available to the collaborator user. On this activity the collaborator user can change a specie information. By doing it, it changes all the captures connected to it, changing their information as well.

The algorithm checks if the specie typed already exists. If it does, the autocomplete feature triggers and, in the case of the user selecting one of the generated from this feature, the rest of the details become automatically filled, as the specie already has those details. If the specie doesn't exist, the user has to introduce all the details regarding it. When this operation happens, the algorithm goes to the initial specie, copies all the information to the new or already existing specie inputted by the user and finally removes the node of the original specie.

Approve/Reject User's Captures

This feature lets the collaborator user accept or reject any new capture made by another user. When a user makes a new capture, it goes to a waiting queue, waiting for approval from a collaborator user. When a collaborator user accepts a capture, it has to fill the information regarding that capture – the specie it belongs to or, in case of a new specie, document it as a new specie and fill the remaining information regarding that new specie.

This feature is tied with the edit capture's information directly. When the collaborator user accepts or rejects a new capture, it has to do the same steps as if it were editing the capture's details. The algorithm is the same.

5.3 Common backend implementation

As explained before, the platform will have two different databases with two distinct purposes. On this section, we will talk about the implementation of the Firebase Realtime Database. This is the main database, the backend that stores and distributes information across all devices.

The use of Firebase Realtime Database grants an easy up-to-date solution throughout all devices. This is a synchronized database, that works by using listeners – functions that listen to changes in the database, warning the applications that changes were made to it. By having this, the synchronization between database and applications is seamless, and mitigates the need for the application to actively look for changes in the database.

Listeners are triggered every time a change is made in the database, fetching all new data for the application automatically. Any change made to the database will have an immediate effect on the data being displayed, without the need to force the application to fetch that new data. The new data is basically given to the application by the database (Figure 31).

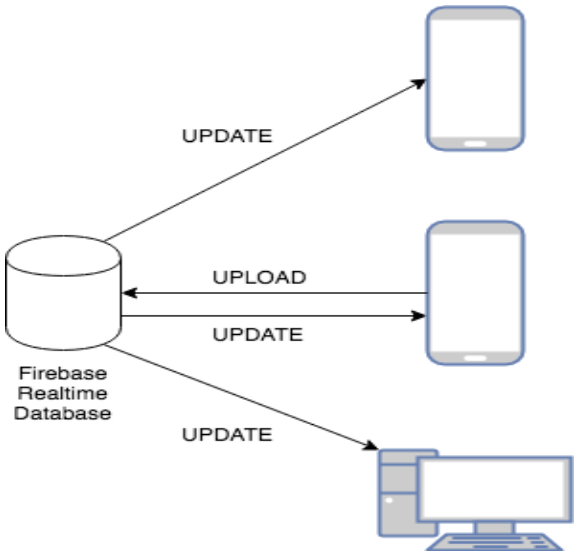

Figure 31 - Firebase Realtime Database information exchange

5.3.1.1 Data structures

All data exchanged is in JSON format. Firebase Realtime Database reads, writes and stores JSON files. Every capture has a determined number of fields:

- Author
- Date
- Ecology
- Position
- Species Name
- UID
- URL
- Vulgar Name

The author and date fields have exactly what the name implies, the author of the capture and the date it was taken. The Ecology field shows the ecology of the capture itself, while the Position shows the location where the capture was taken. Species Name contains the name of the species that capture belongs to; UID is the unique identifier of the user that took the capture; URL has the URL from the database storage that contains the photo and, finally, the vulgar name is the most common name of the specie (Figure 32).

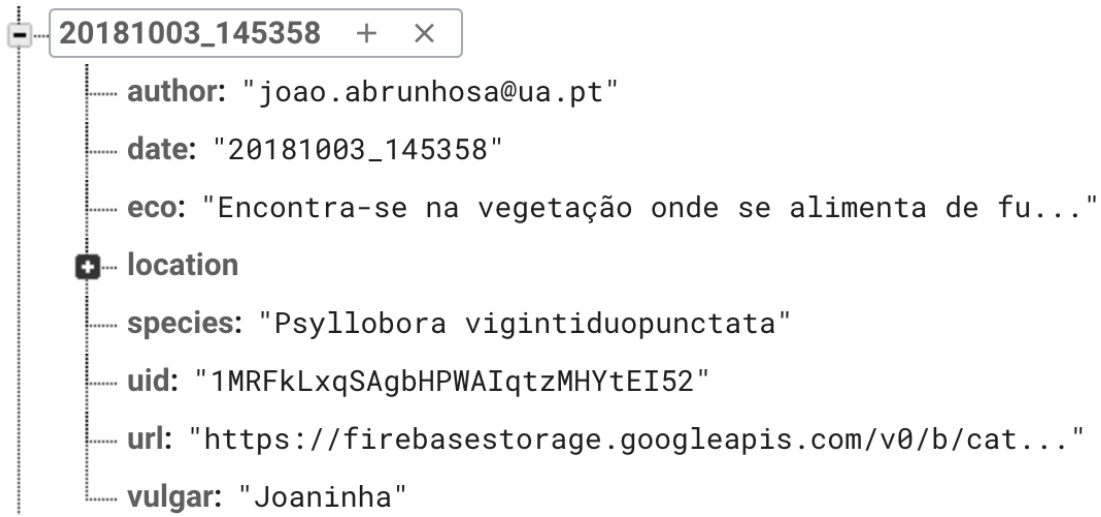

Figure 32 - Individual capture node in Firebase Realtime Database

The use of an URL from the database storage is a solution for the lack of image storage from the Firebase Realtime Database. This database doesn't actually store images or videos so, as a solution, using the Firebase Storage and connecting it to the Firebase Realtime Database through URL was the best approach. The URL is the direct link to the capture that is stored in the Firebase Storage.

Every specie also has a specific number of fields that are common throughout all captures from that specie. Consider the following image:

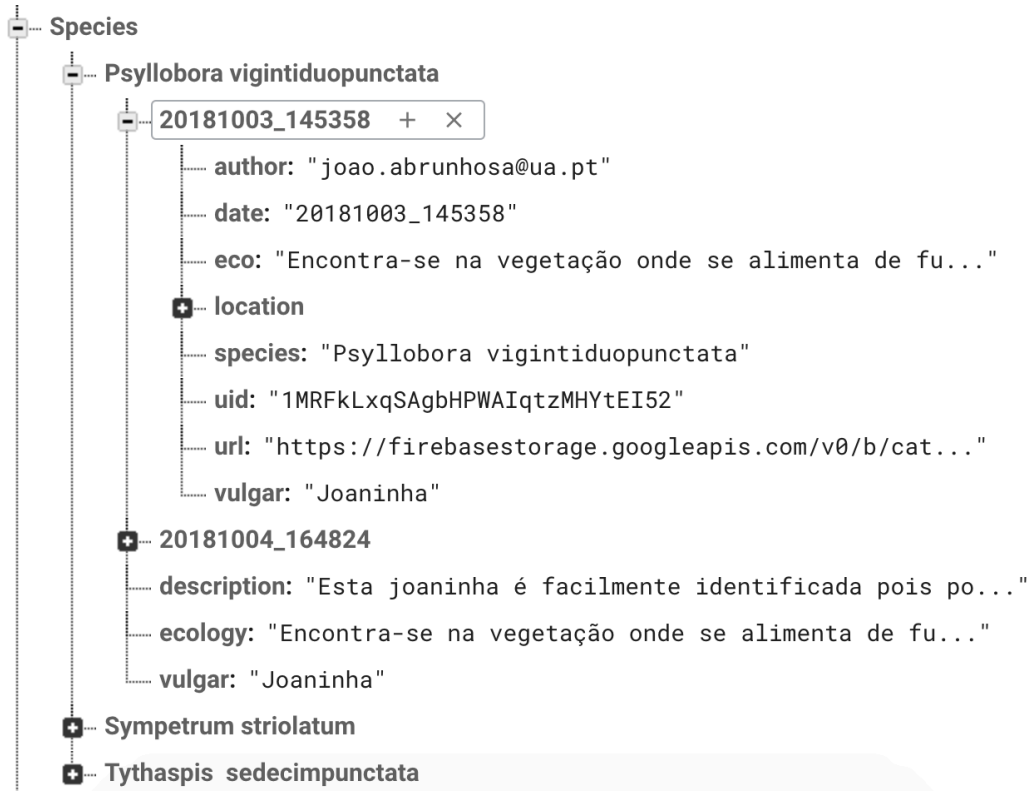

Figure 33 - Species branch

As seen in the image (Figure 33), there are three fields that are common to the specie itself: description, ecology and vulgar. Having these three fields as specie's fields grants a better way of handling information edition, if needed. For example, if a collaborator wrongly documents a specie and it needs posterior changes, by changing only these three fields makes the process much faster. If those fields belong to the capture itself, on the occurrence of said situation, it would be needed to change every capture, making the process slower and affecting the data exchange.

But changing only these three fields is not enough when talking about a complete specie's edition. The name of the specie can also change and that's a problem with Firebase Realtime Database. This database doesn't let the user change the name of the parent node in real time. So, for example, if a user wants to change the name of the "Specie1" to "Specie2", the only way to do this is to create a new entry regarding the Specie2, if it doesn't exist, and copy everything from the Specie1 to the Specie2. This can greatly impact system performance, especially if the specie already has a great number of captures. One by one, the captures are copied to the other node and removed at the same time.

In the case that a specie was wrongly documented and needs to be changed to another already existent specie, the process is much faster. It still copies the captures from one specie to the other, but the information regarding the species isn't changed. Only the captures are moved to the correct node.

5.4 Public web site implementation

One of the main goals of the public web site was to be a hub where information could be shared, such as a blog and an archive. This way, users could keep up with the information regarding the project itself or any information regarding species. For that to work, a Content Management System needed to be implemented. The best and easier to use for the end user is, without a doubt, WordPress. This CMS has a big number of features that lets the user edit information and even the design of the website. Implementing this CMS would make the maintenance of the website easier, without the need of having a big knowledge of web site principles or any kind of programming background.

5.4.1 Architecture

WordPress is a content management system that lets its users tweak and add features, manage content from the website and allows account and blog comments management.

WordPress tool stack with LAMP

The WordPress content management system works on top of another stack of software. This stack is called LAMP – an acronym for Linux, Apache, MySQL and PHP. These are the technologies needed for WordPress to run.

- Linux – operating System
- Apache – web server
- MySQL – database management system
- PHP/Perl – languages used for WordPress

Apache

Apache is a free and open-source cross-platform web server, released under the terms of Apache License 2.0. This platform is maintained by an open community of developers under the Apache Software Foundation.

Apache is what is used to run the web server. This software will be responsible for accepting HTTP connections from clients and replying in HTTP connections to them.

MySQL

MySQL is an open-source relational database management system (RBMS). This platform is the central component of the LAMP open-source web application software stack. It's one of the world's most popular open source databases [23] and enables the creation of cost-effective, reliable, high performance and scalable Web-based applications. MySQL delivers the ease of use, scalability, high-performance and offers a suite of visual tools and database drivers to help developers develop their applications.

On this platform, MySQL will serve as the database for storing any data regarding the WordPress website. Data like images, videos, blog posts, posts commentaries, administrator accounts will be stored in this database.

Developing with WordPress

To develop our own WordPress website, from the source code to its design, the website needed a theme. WordPress offers some standard themes that can be used to make the foundation for a new theme. It's not recommended that these themes source code is altered, so designing a new theme needed to inherit functions from these already existing ones. Basically, a child theme needed to be created.

The parent theme chosen was the "TwentySeventeen" parent theme that already comes with the WordPress package. After that, the pages needed were created and connected to the CMS, so the user can edit those pages at will:

1. The page needs to be created in HTML;
2. The page needs to be replicated in WordPress, with the exact same name;
3. Create a JS dedicated to that page, through the functions.php file.

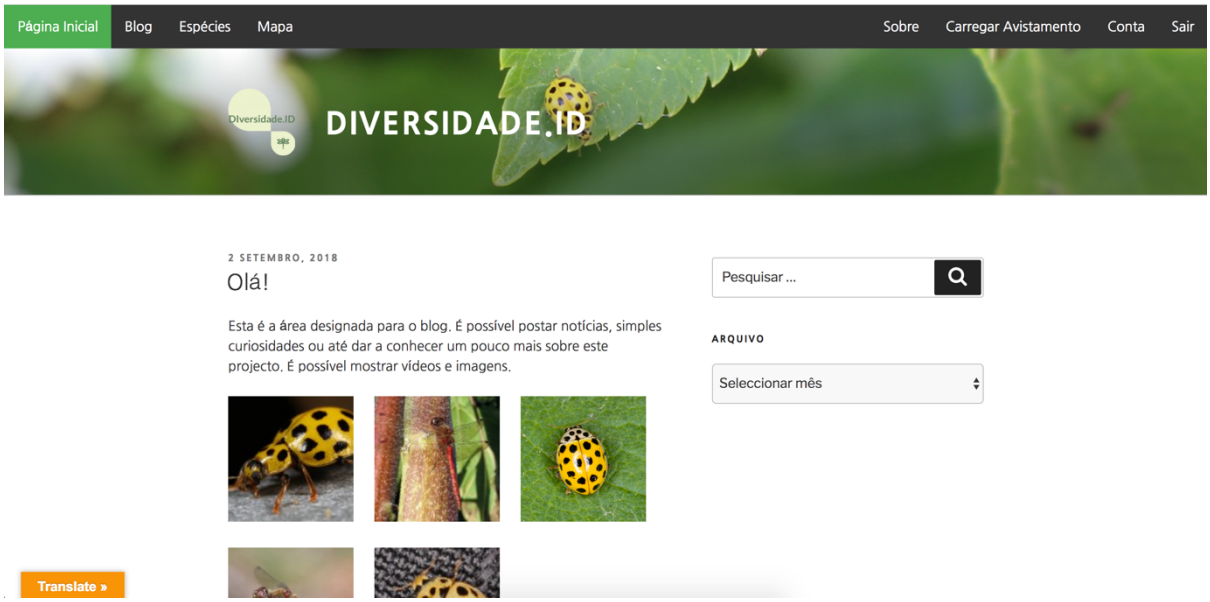

Figure 34 - Blog in the website application

This page (Figure 34), called "Blog", is an example of the page that was created using the parent theme. The design is completely different, but all the functionalities are there, such as the blog.

Consider the following image:

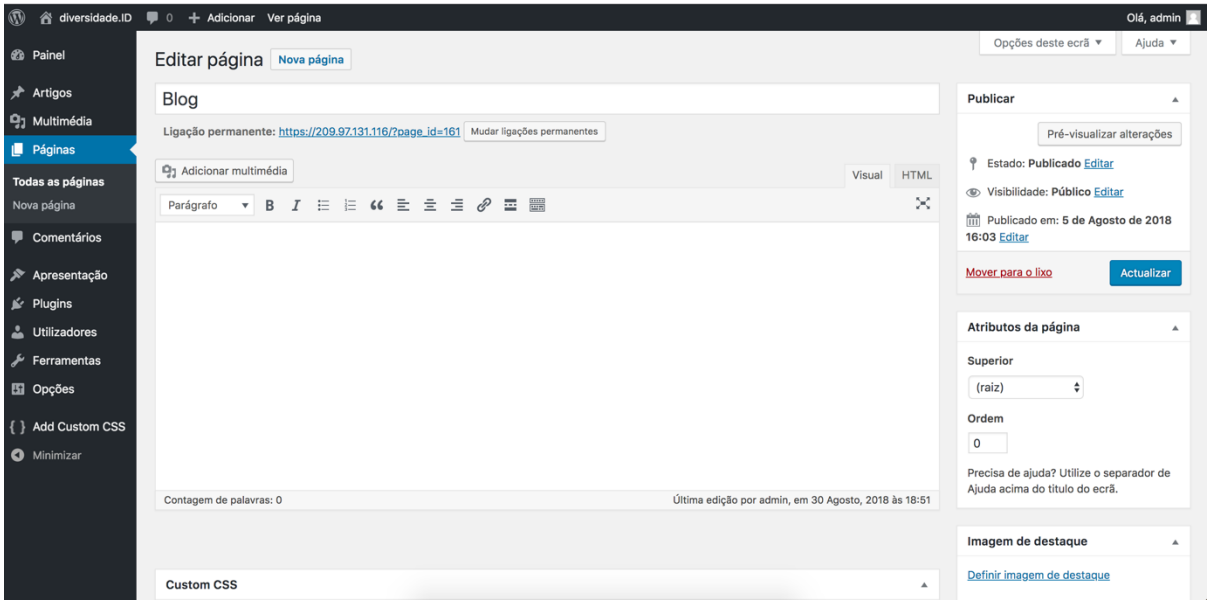

Figure 35 - Page edition in WordPress

This shows how the Blog page can be tweaked to meet the administrator's goal (Figure 35). It can write new blog posts, new articles, put videos or images. It also lets the administrator preview the page before putting the changes public.

Every page has this functionality, and it shows exactly why this CMS was chosen – easy to use, easy to implement, user-friendly.

JavaScript and Firebase Integration with WordPress

The use of JS as the programming language of the website was obligatory thanks to the nature of Firebase - it only works with JS or JS derived framework. So, to use Firebase, JS was needed to develop the website and its integration was crucial to the use of this CMS.

WordPress works based on PHP, therefore not supported by Firebase. This was the first problem faced on the development of the website, but gladly, by using the functions.php file, JS could be run at the same time as the PHP code.

```
if(is_page('mapa')){
  wp_enqueue_script('createMap', get_stylesheet_directory_uri() . '/js/createMap.js');
}
if(is_page('upload')){
  wp_enqueue_script('uploadPhoto', get_stylesheet_directory_uri() . '/js/uploadPhoto.js');
}
```

Figure 36 - JavaScript in PHP

This image (Figure 36) shows how the JS files are loaded. In the first condition, if the page is "mapa" then the createMap.js will be loaded. This condition isn't exactly needed, but without it every time a page would load, all JS files would be loaded, making the page load slower. This solution only loads the needed JS files for the page being loaded.

Another solution was to load the JS files inside the HTML files, but that wouldn't grant that the files would be run at optimal level. The functions.php file runs the JS files in parallel with the PHP files, loading the JS file to the page that the programmer defines.

Databases Distinction and Usage

At this point, two distinct databases have been deployed – Firebase Realtime Database and MySQL.

Firebase Realtime Database isn't the main database for the website, MySQL is. WordPress runs on the LAMP Stack (Linux, Apache, MySQL and PHP) – MySQL works as the main database for the website itself. This database stores blog posts, administrators, commentaries, images, videos – basically all the data from the website. Apache runs the server and PHP is the language that WordPress uses to operate.

So, all data regarding the species and its captures is not stored in the MySQL database. They're all stored in the Firebase Realtime Database. Both databases work in parallel for different functions, on the website (Figure 37).

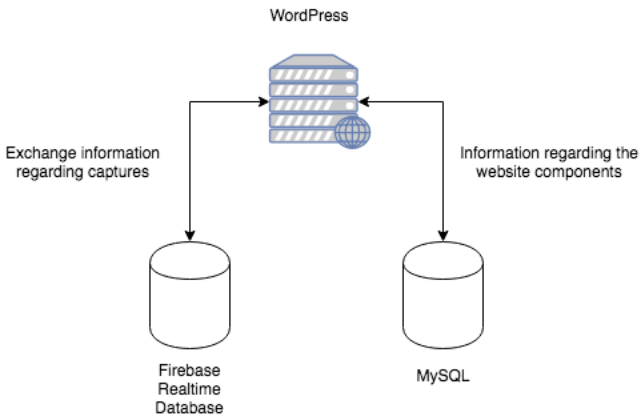

Figure 37 - Difference between the two databases used in the web application

Every time a new capture is made or uploaded, the website uses the Firebase Realtime Database. If the website needs to read something like blog posts, commentaries or images uploaded by an administrator, it reads from the MySQL database.

The application of the Firebase Realtime Database on the website works the same as the Android integration, by using listeners. Those listeners work exactly in the same way as well – every time new data is uploaded, it sends that new data to the devices that are connected to it.

5.4.2 Features

Approve/Reject User's Captures

On the website, the collaborator user can do the same as the mobile application: approve or reject User's captures. Consider the following image:

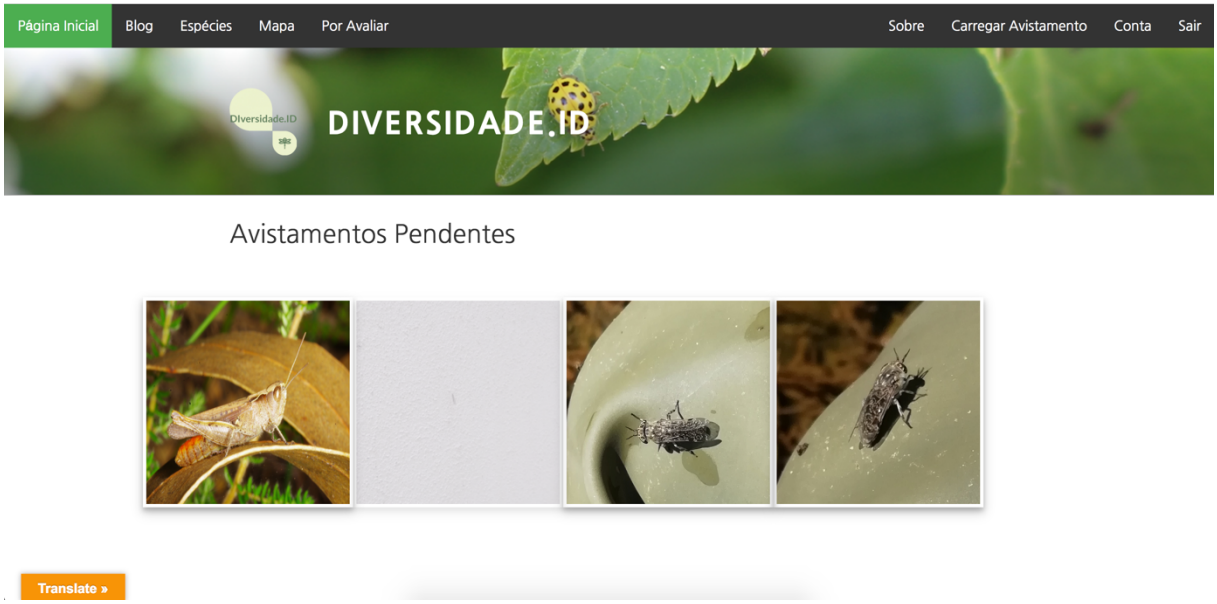

Figure 38 - Pending screen in the web application

By accessing the Pending sightings page (Figure 38), the collaborator user can accept or reject all the captures that appear on the grid (Figure 39). Clicking on the picture shows the following:

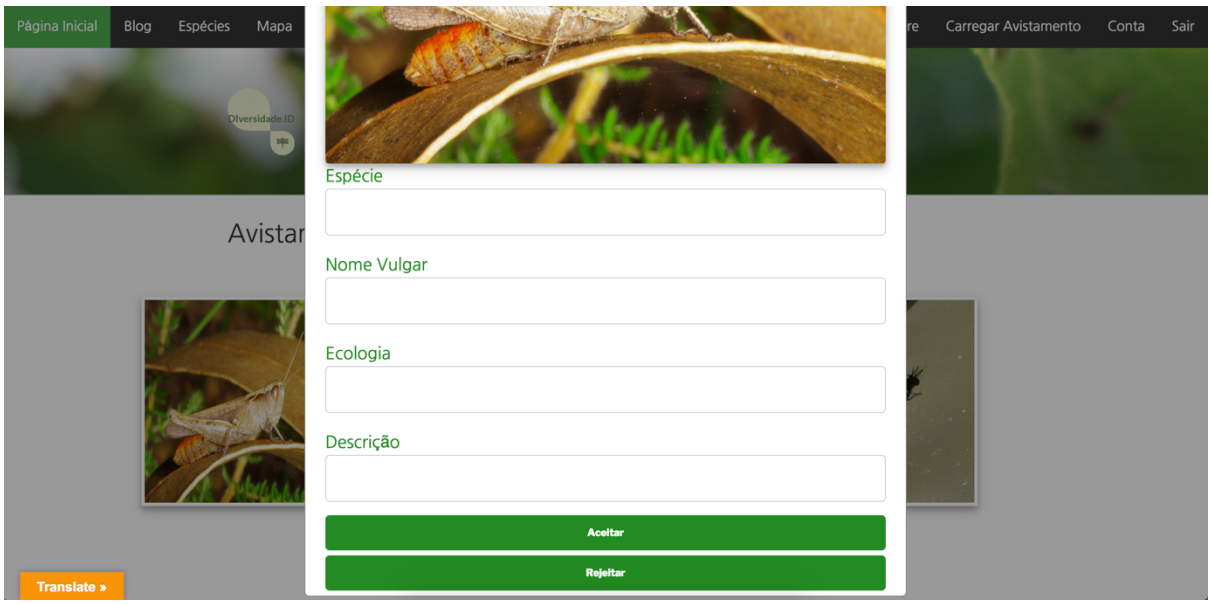

Figure 39 - Capture details edition

The user can add a new specie or evaluate it as an already existing one. If the former happens, then the user has to give the rest of the details, such as ecology, vulgar name and

description. If the latter, then the autocomplete feature will trigger, and by choosing one, the user doesn't need to fill the rest of the details.

This works in the same way as the Android implementation. Initially, the algorithm will add all the species it finds on the database and push them into an array. During the input of text on the specie field, this algorithm will compare the input and see if any specie on the array contains or is equal to it.

After uploading this information, the algorithm will copy the Capture class into the specie chosen and remove it from the pending captures branch

Upload Capture - Collaborator

The user can upload any capture it has saved (Figure 40). Consider the following image:

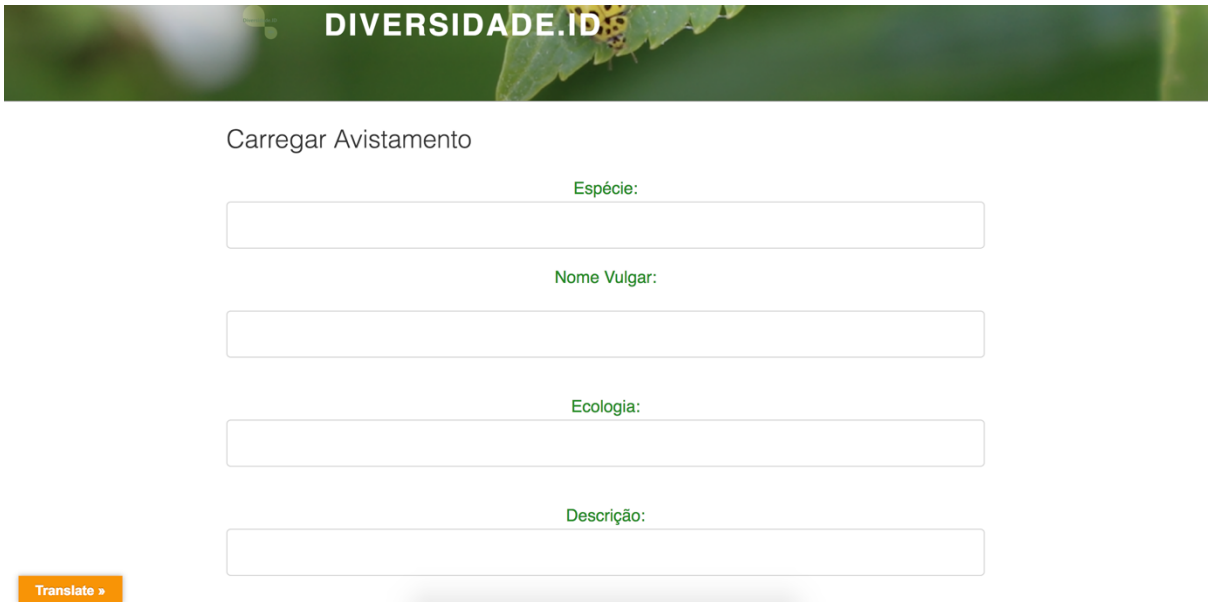

Figure 40 - Upload capture screen (part 1)

Like the previous feature, the user has to fill the fields. If the specie already exists, the autocomplete feature triggers and fills the rest of the fields with the warning that they already exist, disabling the editable fields (makes it non-editable) vulgar name, ecology and description.

The capture also needs a location and the actual capture itself:

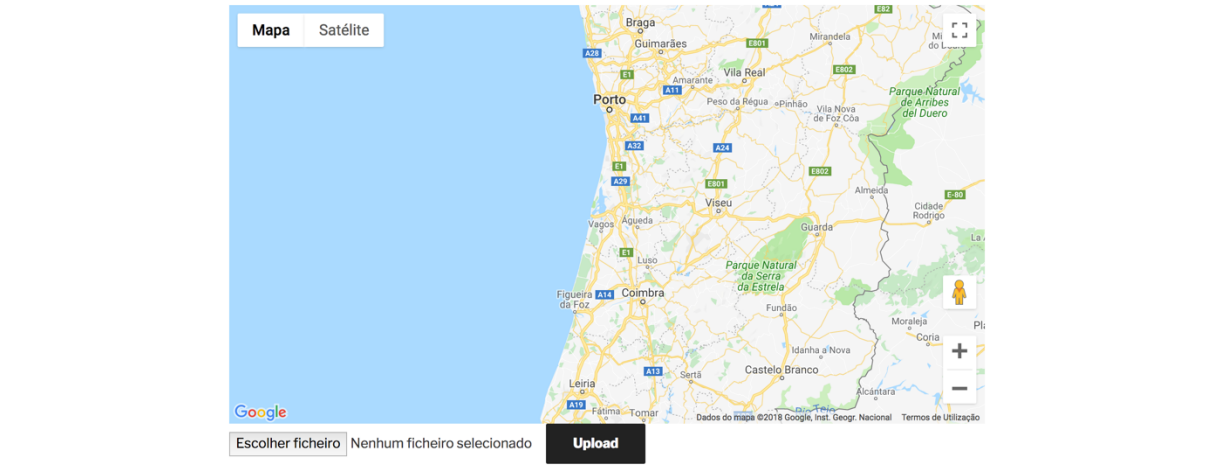

Figure 41 - Upload capture screen (part 2)

For the capture to have a location, the user needs to click on the map where the capture was taken (Figure 41). The last click is always the position that the algorithm will remember as the correct one.

Right below the map it is possible to load any picture from the computer, by using a simple file loader made in JavaScript.

When the user clicks the Upload button, all this information is gathered into a Capture Object class and uploaded that way. It creates consistency on how the data is exchanged between devices. This object doesn't go to the pending captures branch, since it is uploaded by someone who already fills all the necessary information. Instead it goes directly to the specie's branch chosen by the collaborator user.

Upload Capture - Citizen

There is another Upload Capture page, designed specifically for the citizen user (Figure 42). This page lets the user upload any capture it has already saved on the device. The only inputs needed for this upload are the location of the capture and the capture itself.

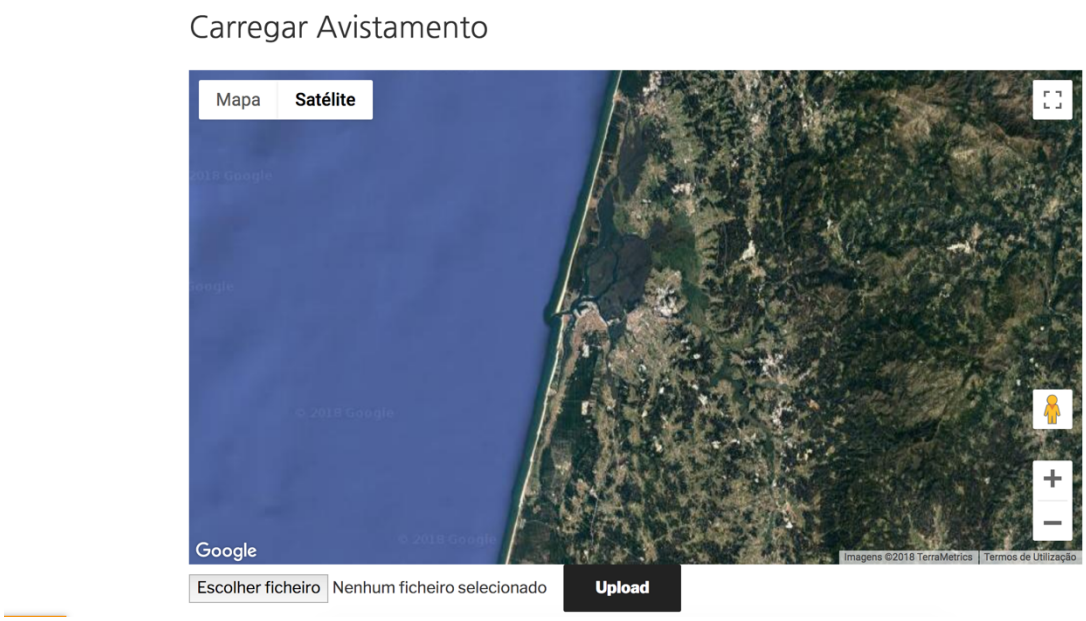

Figure 42 - Upload Capture (citizen)

After the upload button is clicked, the image, just like in the mobile application, is sent to the pending captures queue, where it will wait for a collaborator to document it.

Check Capture Location

To check the capture's location, the user needs to go to the species page (Figure 43) and select the capture:

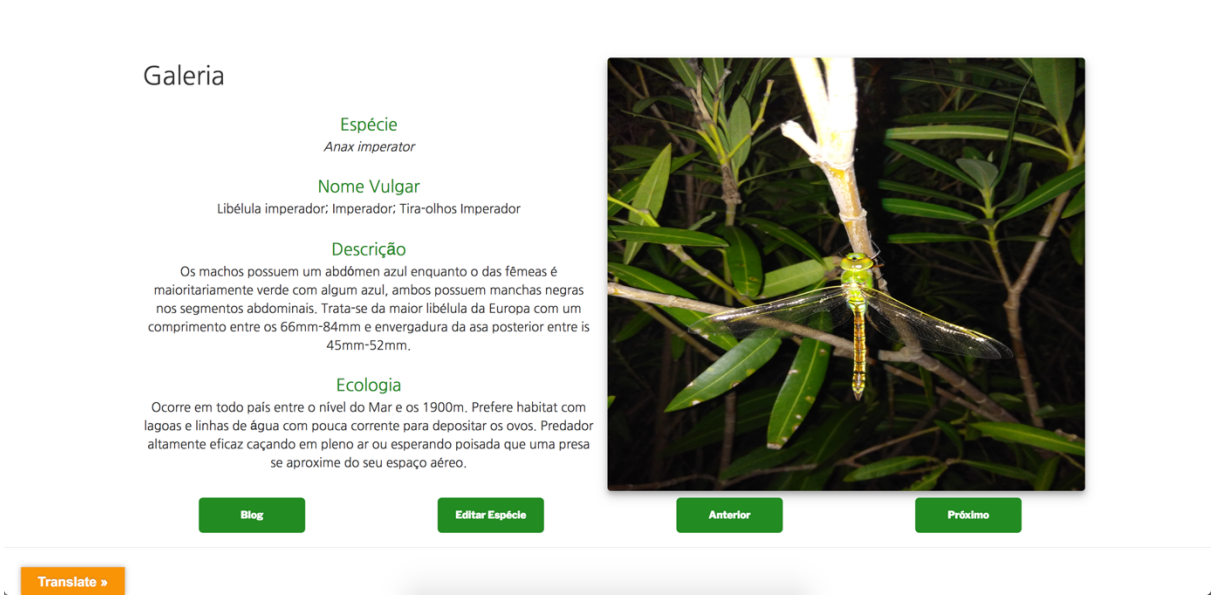

Figure 43 - Specie's page (Anax imperator)

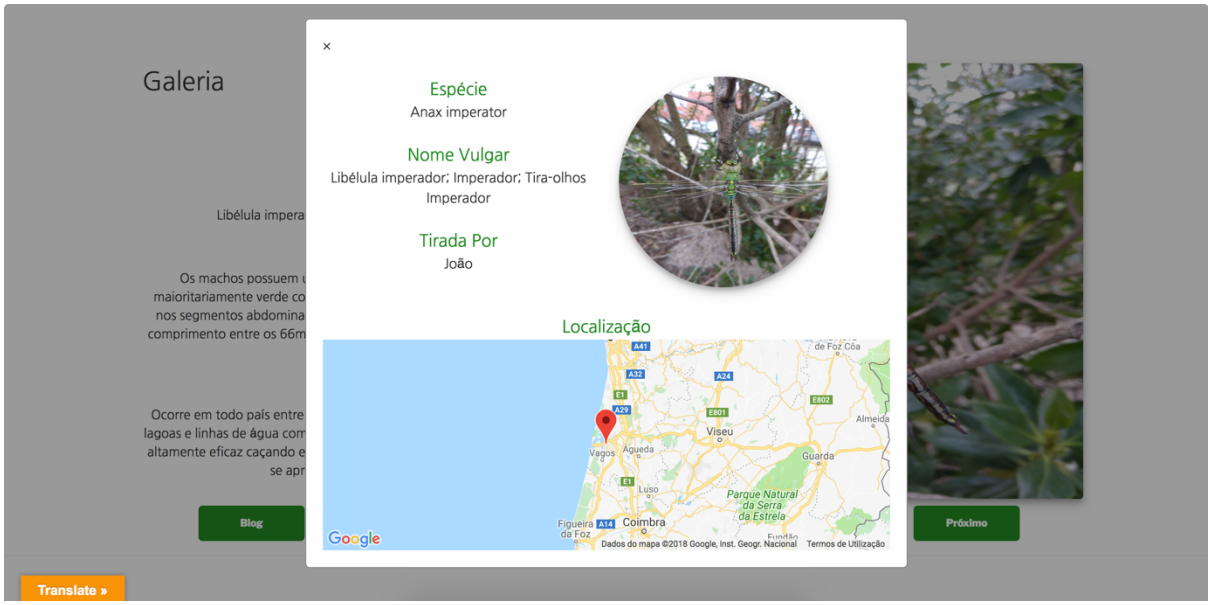

Figure 44 - Capture information

When the user clicks on the capture, a pop-up appears showing the specie's name, the image clicked, vulgar name, who took it and the location (Figure 44).

The captures details are all gathered when the page is loaded. An image array is created, containing all the information regarding all captures of the specie. When the user clicks on an image, all the information is ready to be shown.

Check Species Location

To check the species location, the user needs to go to the "Map" page (Figure 45). When the page is loaded, the user will be presented with the following screen:

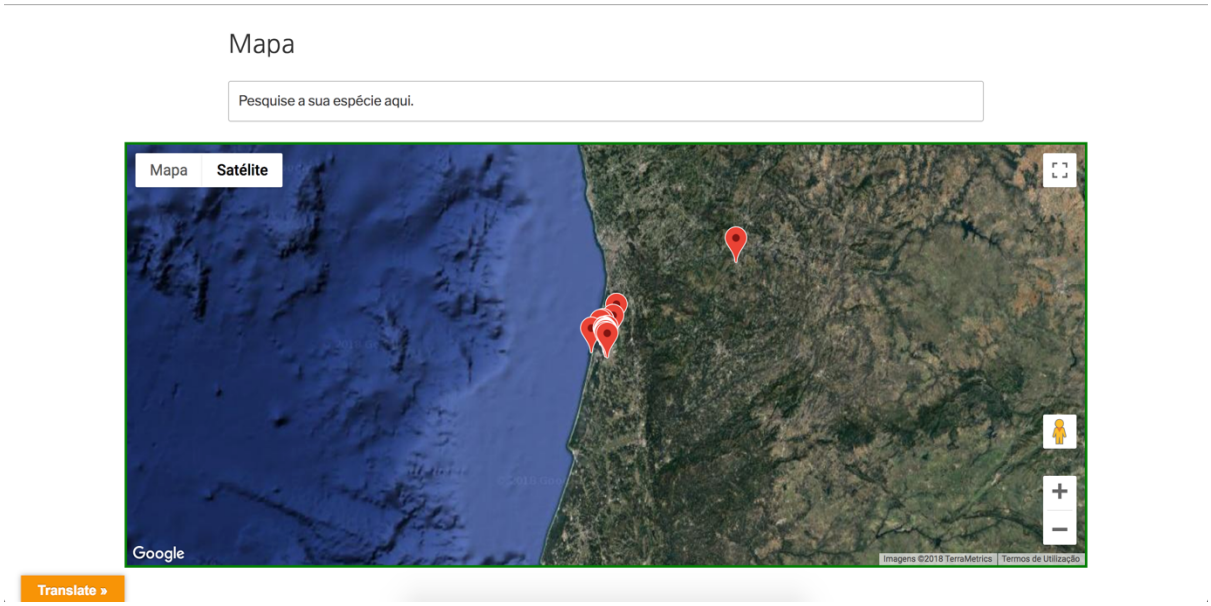

Figure 45 - Map screen

A search bar is also present, so if the user wants a certain specie's location, it will be easier to watch it on the map as it only shows the specie searched by the user.

Once again, the search bar will have a autocomplete feature that will trigger every time the user searches something.

This autocomplete feature is done by adding all species to a list every time the page loads. As the user types its input, the search function will constantly compare the value read to all the values in the list. The algorithm cycles through all species, saves the captures position and its name, and stores it in an array. If the user doesn't type anything, then it shows all the species.

Check Species Details

Every user can check any species details. First, they click on the "Species" tab (Figure 46):

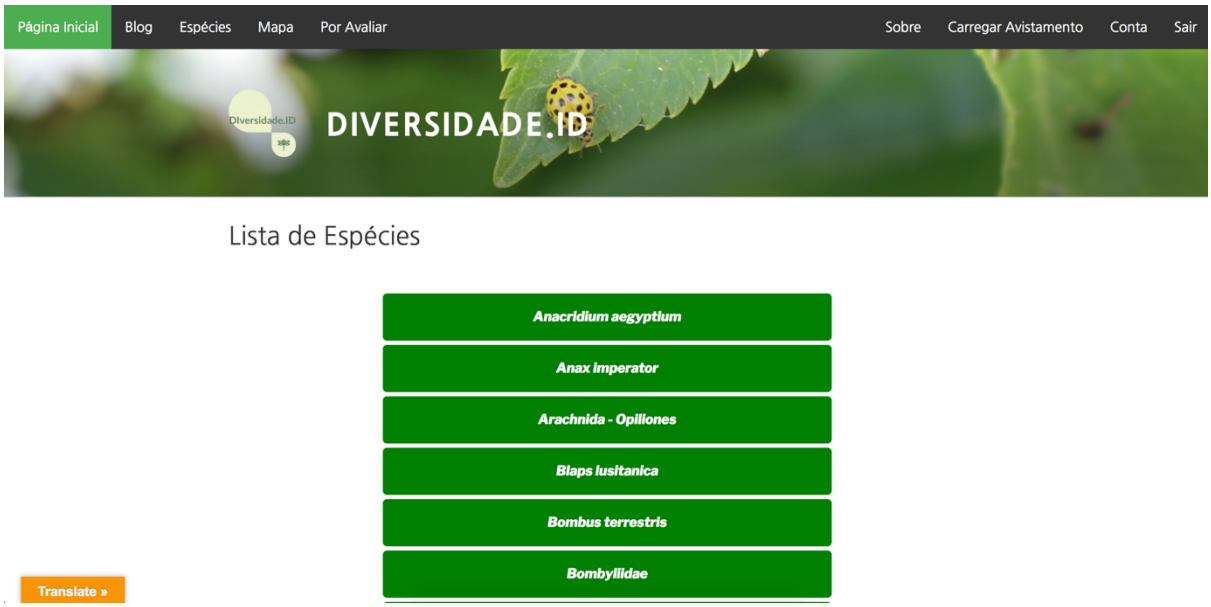

Figure 46 - Documented species list screen

Clicking on one of the species leads the user to that specie's page:

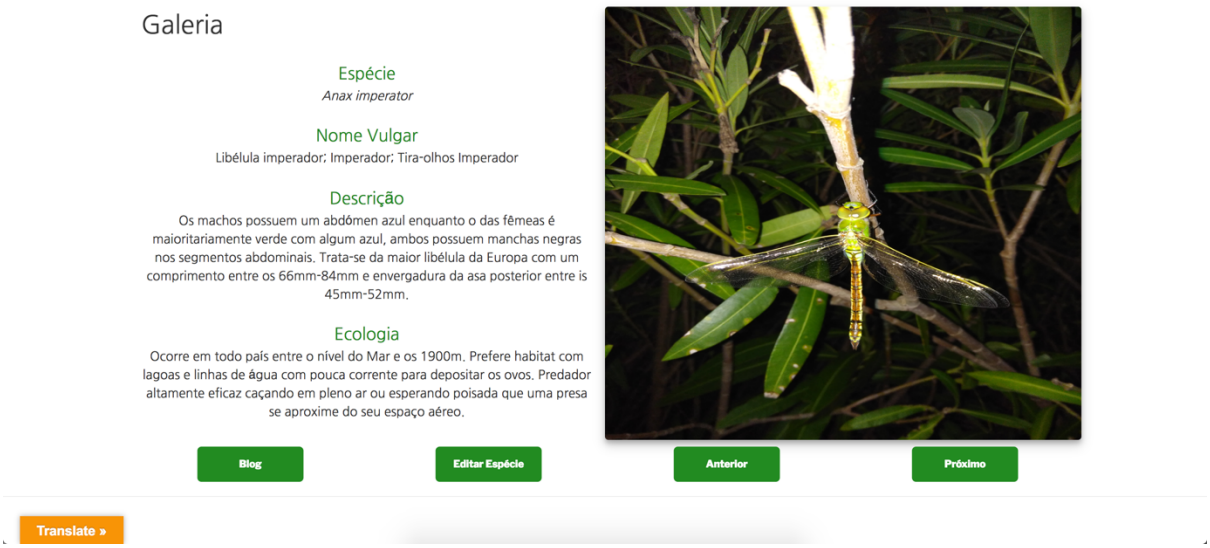

Figure 47 - Species information screen (Anax imperator)

As seen on the image (Figure 47), it has four buttons: Blog, Edit Species, Previous and Next. The first one leads the user to the blog posts regarding the specie selected; the "Edit Specie" button only shows to the collaborator user and lets it edit the details of the specie; the "Previous" and "Next" buttons cycle through the gallery of captures.

The algorithm gets the current specie being viewed and uses it to get the information it needs from the Firebase Realtime Database. After getting the information, it simply maps the information to the correct elements.

Clicking on the "Edit Specie" button, the user is presented with the following (Figure 48):

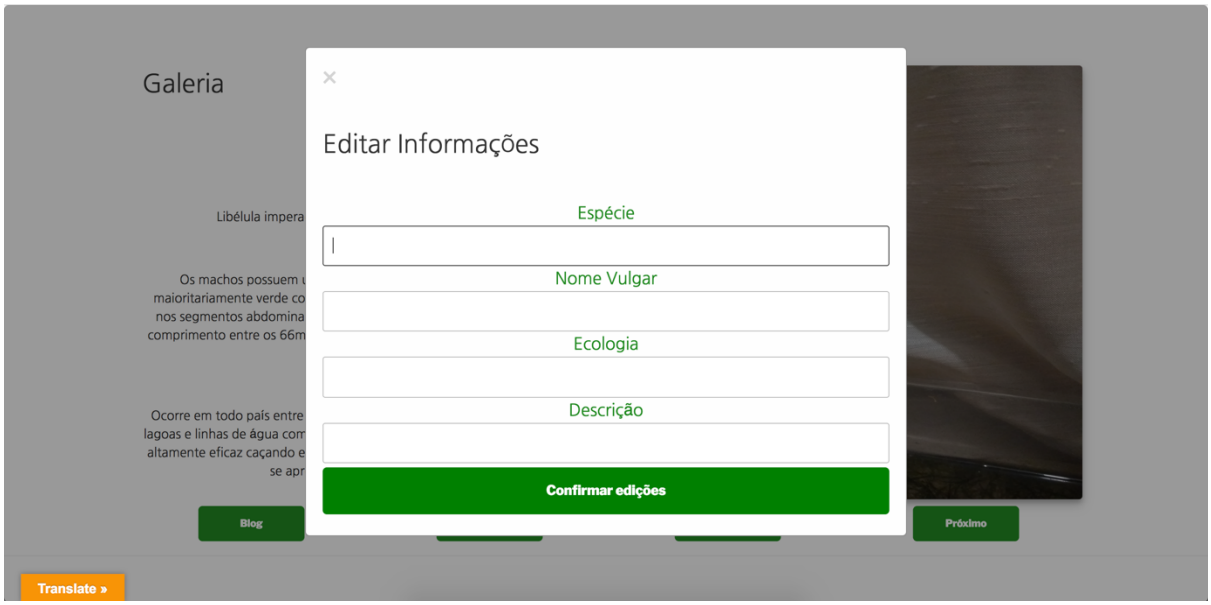

Figure 48 - Specie's information edition

This works the same way as the Android implementation. If the specie changes, then the information and all the captures are sent to the new or already existing specie and removed from its original one.

Blog

The blog works as an information hub, where the administrator can post new articles, images or videos, and news (Figure 49). Users can comment, comments that are moderated by the administrator itself. The administrator can accept or reject a comment.

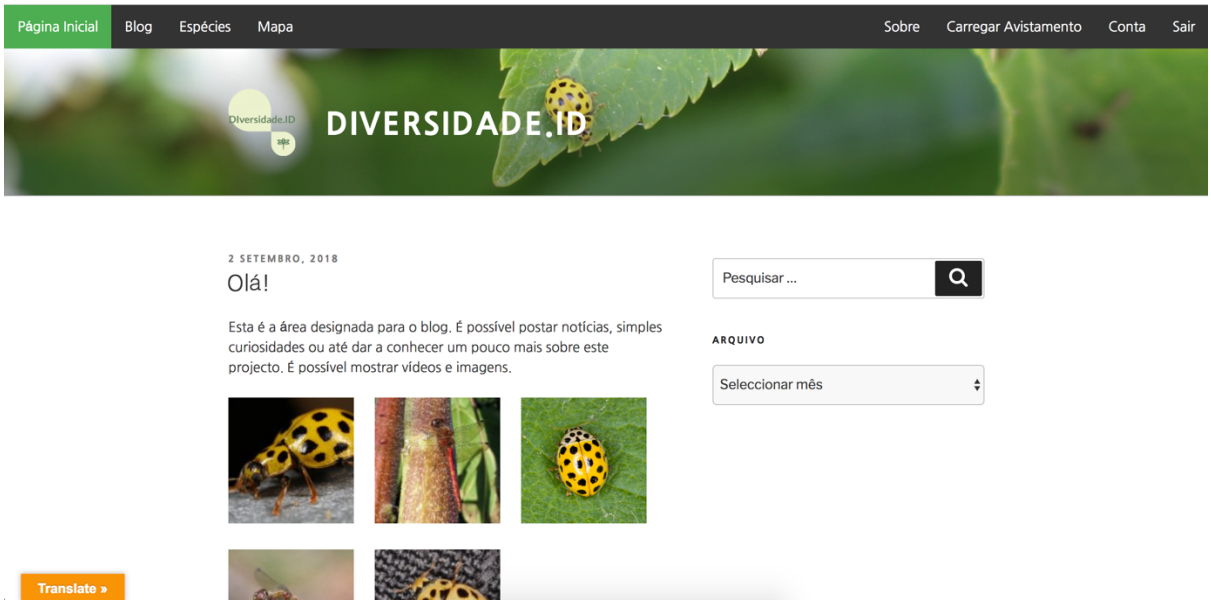

Figure 49 - Blog screen

Clicking on the blog post will lead the user to a more detailed view of the post. The user can comment, but it has the obligation of using the email to comment, by default (Figure 50).

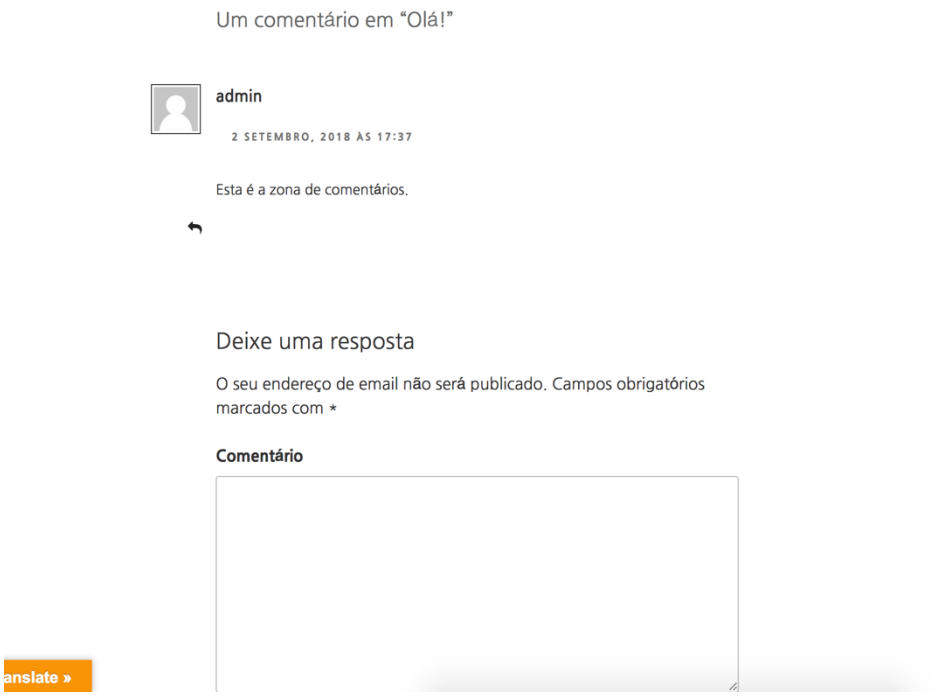

Figure 50 - Comment area in a blog post

After a comment is made, the administrator has the power to accept or reject such comment (Figure 51). This can also be changed by the administrator, for example, revoking the need of using an email to comment on the blog post and to accept the comment.

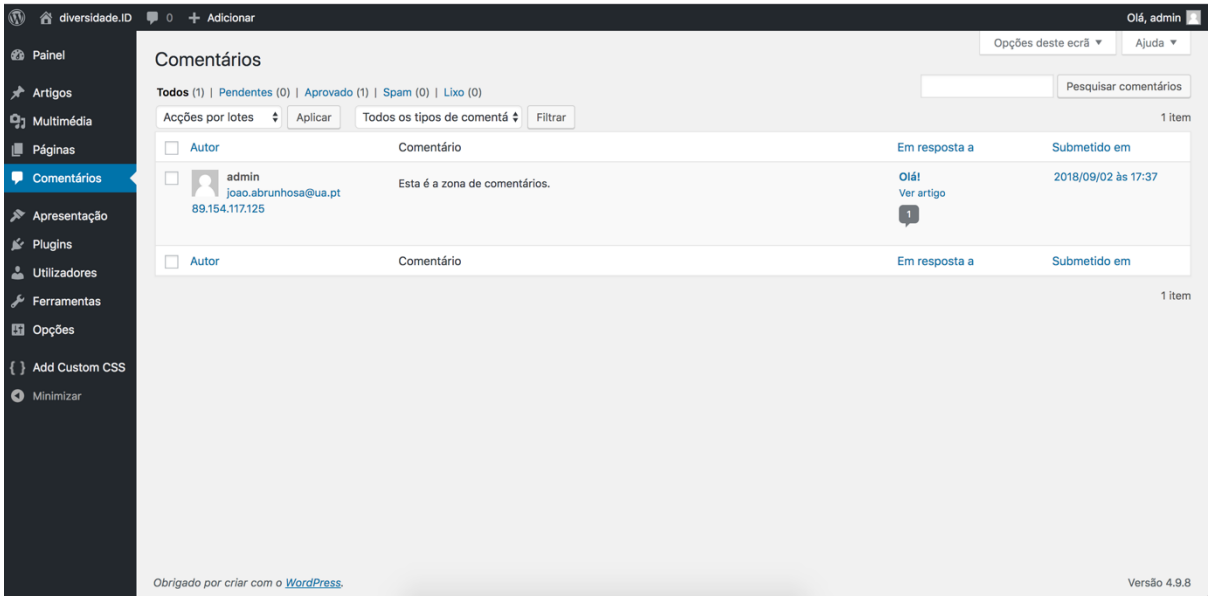

Figure 51 - Comments management in WordPress

5.5 Cloud deployment

For the website to be available to everyone, it needs to be deployed into a public server that can be accessible from everywhere. At first, a local web server was the chosen approach for this problem, but time constraints and maintenance costs are big disadvantages for the deployment of this local web server. To avoid this time and costs problem, a cloud web server is the solution: it gives us the option to have a customized web server hosted on another location, without worrying about the maintenance of the web server.

The web server chosen had the following specifications:

- Ipv4: 209.97.131.116;
- RAM: 1GB;
- Disk: 25 GB SSD.
- Location: London, UK

This gives us the sufficient power to run WordPress and its components. The virtual machine chosen was a LAMP based VM, running on Ubuntu 16.10. The LAMP stack, like already explained, is a stack used for WordPress development. It stands for Linux, Apache, MySQL and PHP – all software needed to run WordPress.

Since the VM was already running Linux, there was no need to install a Linux client, so the next step would be to install Apache, MySQL and PHP.

After the installation, all needed is to download the WordPress files and configure the files to the correct paths of the Apache and MySQL directories.

5.6 Quality assurance

To guarantee that the mobile application would run correctly, a variety of tests were deployed, using Espresso. On this chapter we will show and discuss the results for the quality assurance tests.

Login

For the Login test, an example email and password were put as input for the two fields. After the input, the algorithm clicks on the Login button.

Register

This test was the same as the Login text – the algorithm fills the fields necessary for the registration process to be completed. In the end, click on the register button.

For this test, two distinct ones were created for the citizen user and the collaborator user.

Check Capture Details

To test the capture details, the algorithm first logs in into the platform and immediately clicks on a capture from the image feed, if it exists. After that, it checks the location of the capture.

Check Species Details

To test the species details, the algorithm logs in, goes to the species guide activity and clicks on the first one. After that, it checks the species location and scrolls down to see the captures related to the species.

Check Species Map

After logging in, the algorithm goes to the Map activity. There it searches for a specie and clicks on it.

Edit Capture Details

As always, the algorithm first logs in, goes to the pending captures activity, and clicks on the first one. It's then redirected to the capture details activity, and there it clicks on the edit button. When in the edit details activity, the algorithm fills the specie's field with a new specie. Because it's a new specie, the algorithm fills the rest of the details and clicks on the accept button.

Edit Species Details

First the algorithm logs in, then it goes to the species guide activity. There it chooses the first one and it's redirected to that specie details activity. It clicks on the edit details button and fills the information for a new specie. Finally, it clicks on the accept button.

Results

Unfortunately, due to time constraints, the test could only be run on two devices. They were made manually, using the Espresso test recording.

Task	Huawei P7 Mini (API 19)	Huawei P9 Lite (API 24)
Login	Success	Success
Register	Success	Success
Check Capture details	Success	Success
Check Species details	Success	Success
Check Species map	Success	Success
Edit Capture Details	Success	Success
Edit Species Details	Success	Success

Table 4 - Espresso tests

Some tests are missing – Take a capture or Upload capture. These tests could not be completed because of how the Camera API and Espresso works. Espresso only works within the application and calling the Camera API takes the user out of the application. Using the camera with Espresso is not possible and the only work around this is by mocking the camera intent. Due to time constraints this was not possible.

6 Results and validation

6.1 Prototypes and supported interactions

The final application and website can execute what was designed during the initial states of prototyping. All the features could be implemented, and on this chapter those features will be described and explained. The following list shows the supported features of each the mobile application and website:

- Mobile Application:
  - New Capture
  - Edit Capture
  - Edit Existing Specie
  - Check Capture Location
  - Check Specie Location
  - Accept/Reject captures from citizen users
  - Check other user's photos
  - Check Capture Information
  - Check Species Information
- Website:
  - New Capture
  - Edit Existing Specie
  - Check Capture Location
  - Check Specie Location
  - Accept/Reject captures from citizen users
  - Check other user's photos
  - Check Capture Information
  - Check Species Information

New Capture

This feature is available on both platforms. It lets the user make a new capture and upload it to the queue to be documented by a collaborator. Both collaborator and citizen user can do it, with the difference that the collaborator has the option of documenting his own capture, while the citizen user has to wait for the approval and documenting of the collaborator user.

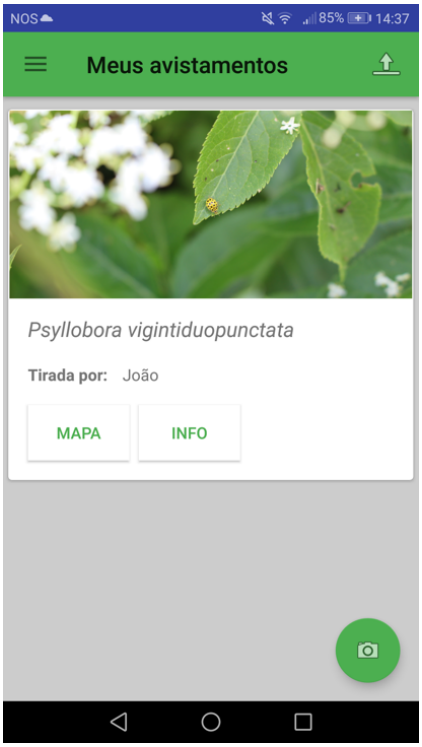

Figure 52 - User's captures gallery feed activity

On the mobile platform, the user can make a capture, independent on where it is in the application (Figure 52). That feature was designed for the user to quickly capture an insect, without the need to go to a specific activity. Doing so, it opens the camera app of the smartphone, calling the Android Camera API. After taking the picture, the user is given the option to accept or reject the current photo and, if accepted, the image is uploaded to the Firebase Realtime Database.

Edit Capture & Edit Existing Specie

The Edit Capture feature is only available on the mobile application. This lets the collaborator user edit any photo that it's already documented or yet to be documented. In a way, it's connected to the New Capture feature.

Figure 53 - Capture information activity

As seen in the image (Figure 53), the user can edit the capture by clicking on the button. That button is only available to the collaborator user and clicking on it leads the user to the following activity:

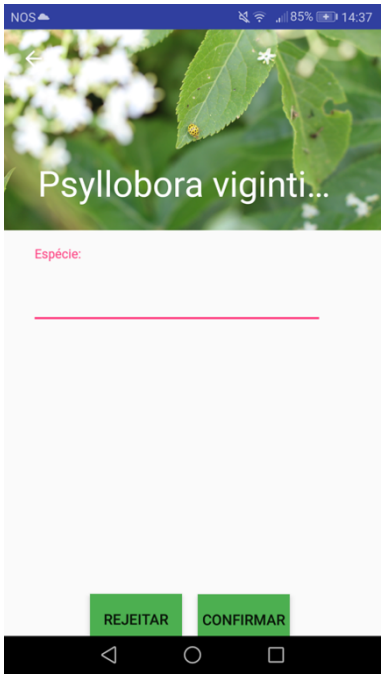

Figure 54 - Edit capture's details activity

The user can change the details of the capture, from species to the description (Figure 54). If the user wants to change the specie, then it can add the capture to an existing specie or create a new one. In the case of wanting to add the capture to an existing specie, the autocomplete feature triggers, letting the user choose the correct one. The description,

ecology and vulgar name are automatically filled. This saves the user from typing long names and making the whole process easier (Figure 55).

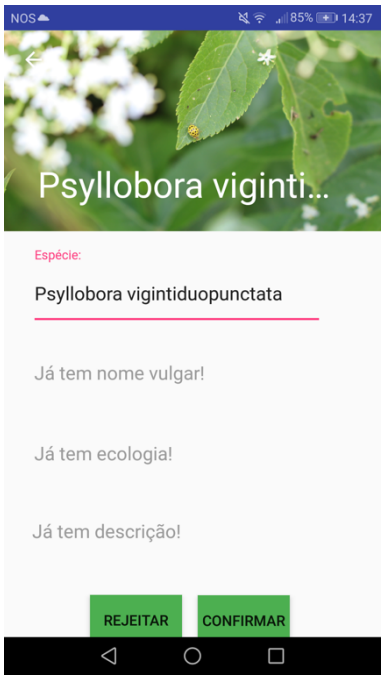

Figure 55 - Edit capture's details activity with an already existing specie

In the case that the collaborator user needs to edit an already existing specie, that option is also available. In the case of changing a specie to another that already exists, the same functionality is present here: autocomplete triggers, the user chooses one, and the rest of the details are automatically filled. In the case of being a new specie, then the process is about filling all new information regarding the new specie.

Check Capture and Species Location

If a user wants to check the capture's location, it needs to go to the capture's page and click on the button with a map on it. By clicking, the user is redirected to a map fragment that shows where the capture was taken (Figure 56 and 57).

To check the position of all species, the user needs to go to the Map activity. There all species are show and they can be searched, in the case of the user wanting to find a specific one. Using the search bar pans the camera automatically to the specie the user is looking for (Figure 56 and 58).

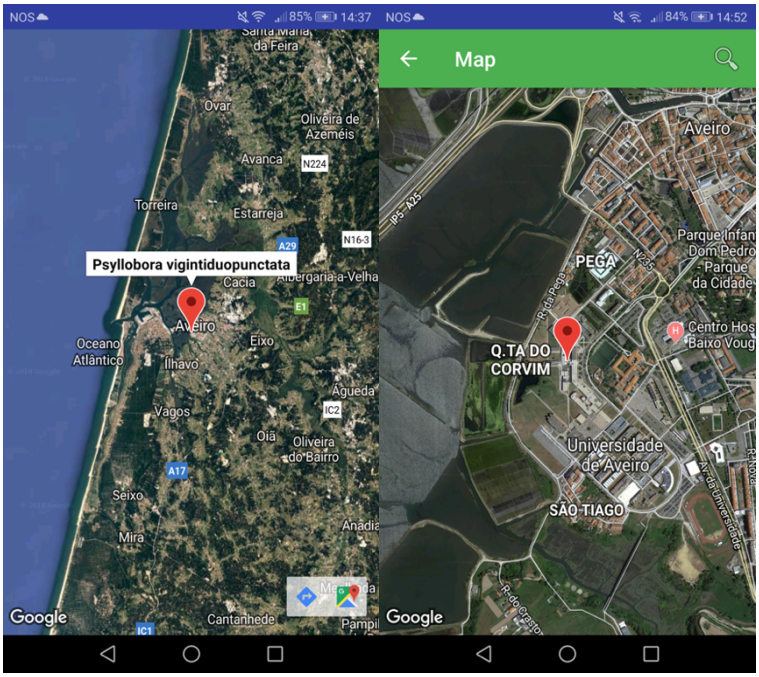

Figure 56 - Capture location (left) and Species locations (right) on mobile

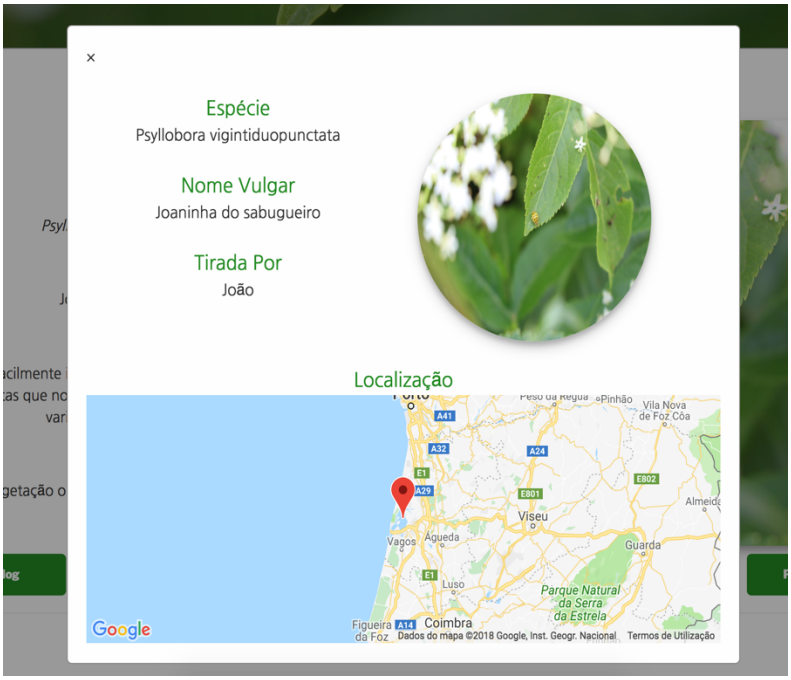

Figure 57 - Capture's location (website)

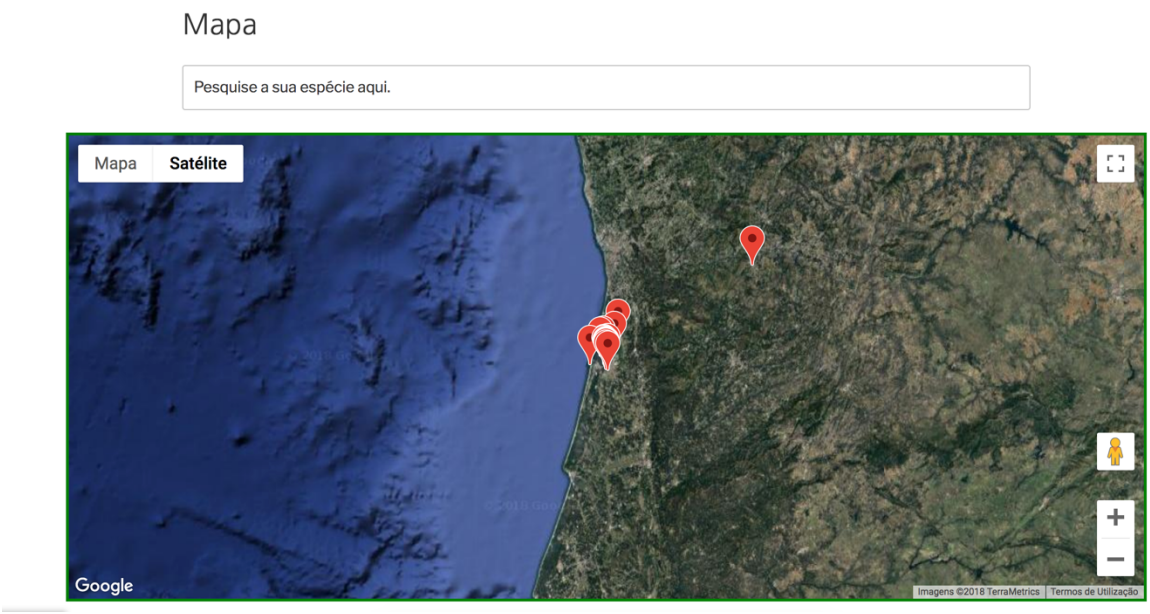

Figure 58 - Species location (website)

Accept/Reject captures

The collaborator user can accept or reject any new captures made by other users. Notifications are sent to the collaborators that a new capture was taken and it's waiting for approval. When the collaborator user clicks on the notification, it's redirected to the "Pending Captures" Activity, where it can choose which one of the captures to document.

By clicking on the capture, the user is redirected to the capture activity that contains all information, and it's able to change the details of that capture (Figure 59).

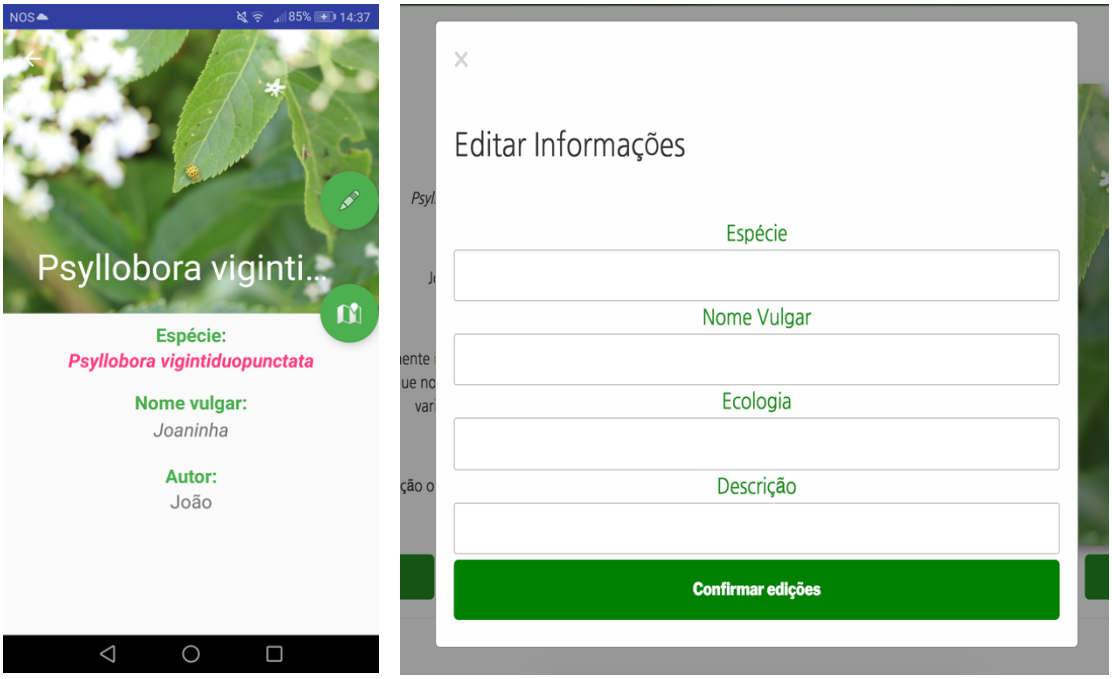

Figure 59 - Edit capture's details activity (mobile and web)

The details are put in a Capture Object, through the setters, and uploaded to the Firebase Realtime Database. There, the Firebase will replace the already existent entry of that capture with the new one with the details.

Check other user's photos

Every user, collaborator or not, can check other users captures. When they go to the "Capture Gallery" activity, they can see every picture taken by other users (Figure 60). They can check the details and the location of the capture.

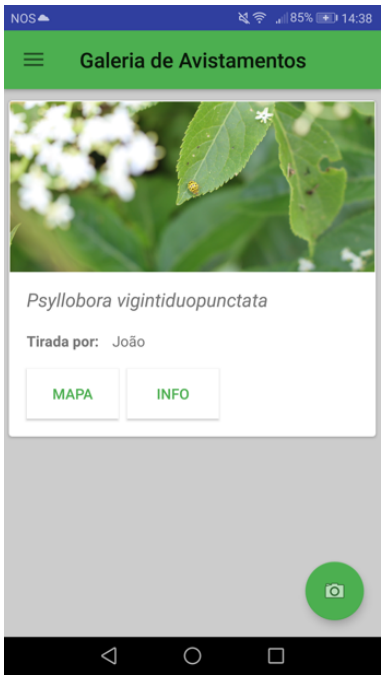

Figure 60 - Captures gallery feed activity

Check Capture and Species Information

By clicking on a capture, the user can check the information regarding that capture. Details like the author, specie of which that capture belongs to and the location of the capture. Clicking on the specie's name on that same activity, leads the user to the specie hub, where it can check everything regarding that specie such as name, vulgar name, description and ecology. It can also check the location of where that specie can be found (Figure 61 and 62).

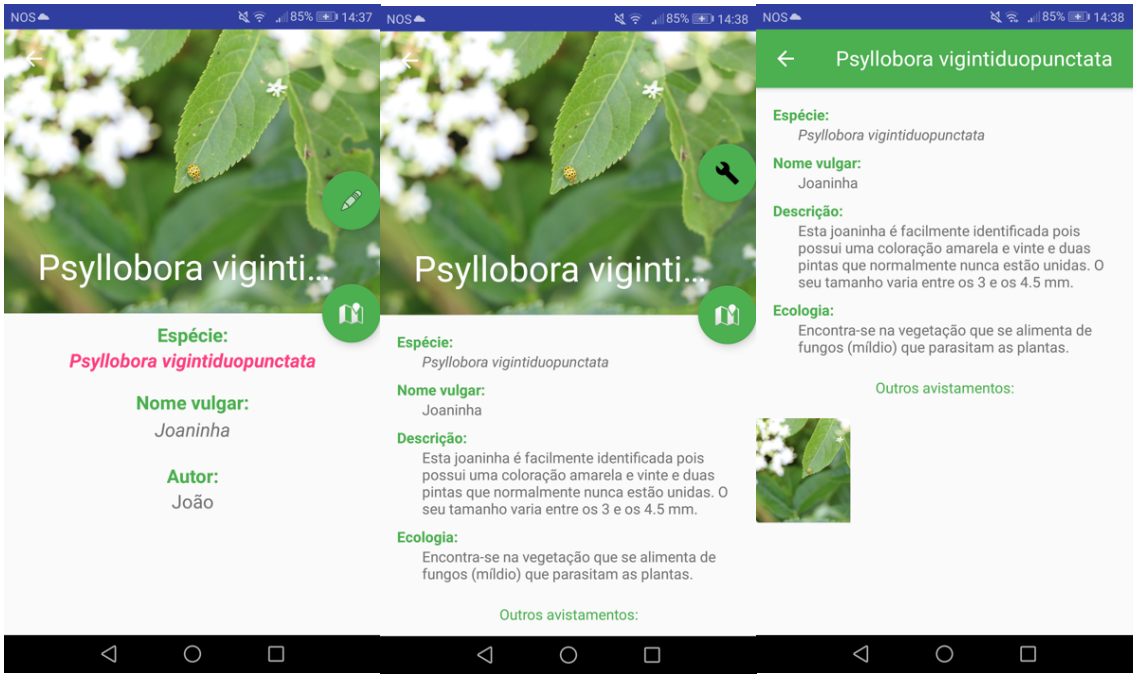

Figure 61 - Edit capture details (left), specie information activity (middle and right)

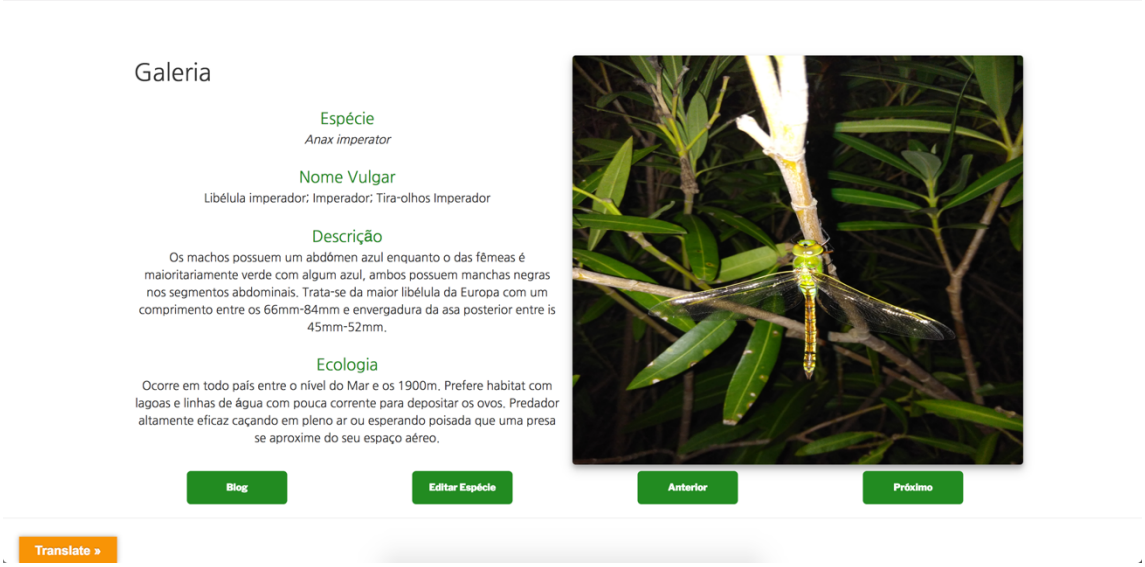

Figure 62 - Species details (web)

6.2 Usability review and testing

For both the application and website to be usable for everyone, independent of age and knowledge of any kind of technology, they should be easy to understand and clearly define what the application and website are supposed to feature.

The current design of the application and website went through increment changes, changes that were made to support new features as they were being included. The initial design drawn during the prototype phase and the final design aren't that much different: the main pillar for the design kept the same throughout all the development process.

On this chapter, first we're going to explain the final design and the choices made with it and last the usability testing made to evaluate if the design was suitable for the goals.

Design choices – Mobile Application

First, the main goal of the mobile application was to look like some well-known image feed-like applications, such as Instagram, for example. Since this application is a way for users to check their and other users photos, an image feed was the most suitable. Consider the next image:

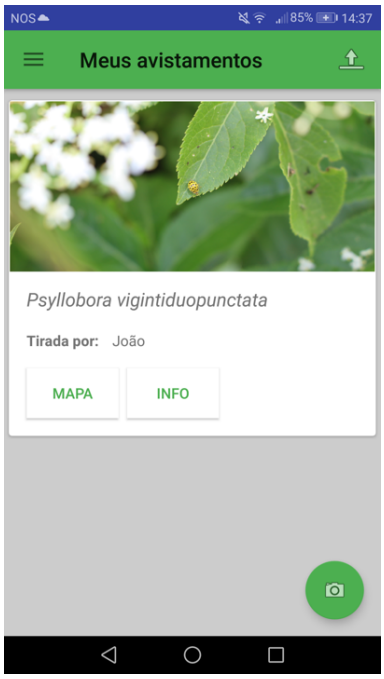

Figure 63 - User's captures gallery feed activity

As seen in the image (Figure 63), the main pillar for this design is still present: the image feed. But since this image feed is only the top layer of what really matters to the user, it's needed to let the user know that there's more to the image, like the position where it was taken and information regarding that capture. So, putting those two features on the container shows that to the user. The user can't be left wondering if there's more to it or not – it needs to be told that it exists.

There are three image feeds on the mobile application, and all of them work the same way- image and two buttons (Figure 64).

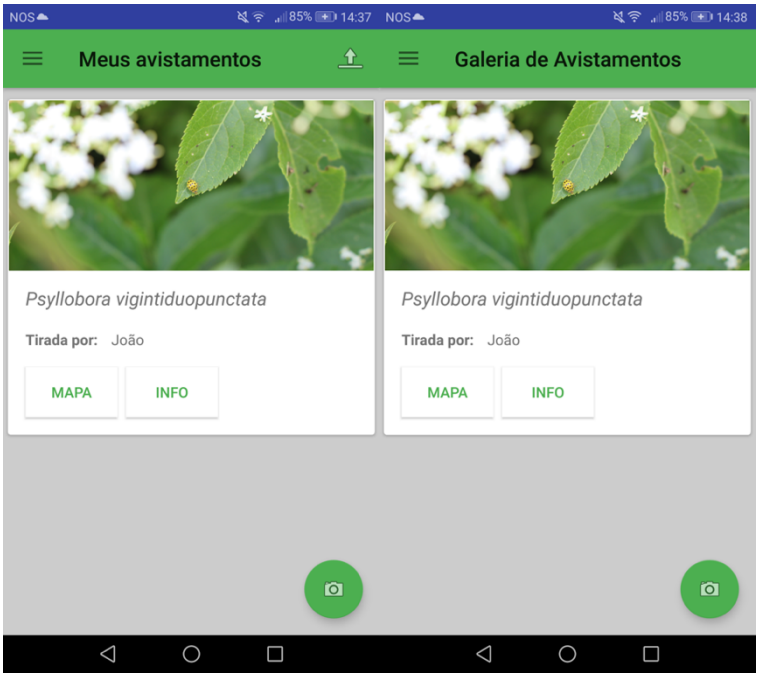

Figure 64 - The holder is the same in both activities

Next, the capture's information activity (Figure 65). This activity should show the basic information regarding that capture, such as the author and what species it belongs to. It can't show every information about the specie – that should be done in a central activity that gathers all information regarding that specie. It can also be possible to check the location of the capture, by using the Floating Button with a map icon. It's a simple way to tell the person that the position is available to check, just by clicking on that button.

Figure 65 - Edit button on the right side of the image (top) and map location button (bottom)

The specie name is in another colour, showing the user that more information regarding that specie can be checked by clicking on the name. Doing that will redirect the user

to the page regarding that specie (Figure 66). If the capture has no specie, then it won't have any effect if the user decides to click on that text view.

Figure 66 - The user is redirected to the specie information activity

The floating button with the image of the pencil is only visible for the collaborator user (Figure 65). This lets the collaborator know that the capture is editable, and any changes can be made to it (Figure 67).

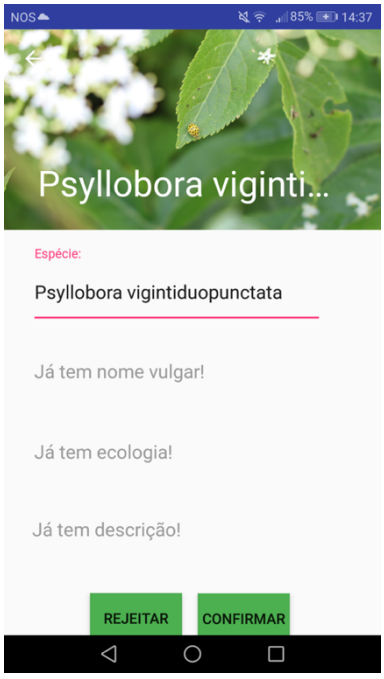

Figure 67 - Capture information being edited

Another important activity that helps the user with the gathering of information of species is the “Species Guide” activity (Figure 68). This activity shows all the species documented and has a search function designed to help the user find the specie that it wants. The user can search the specie by their vulgar name or by the scientific name. Using the vulgar name will, for sure, be most common around citizen users with no actual knowledge of the entomology field, while the scientific name is more precise and will let any collaborator find the specie it wants quickly. Consider the following image:

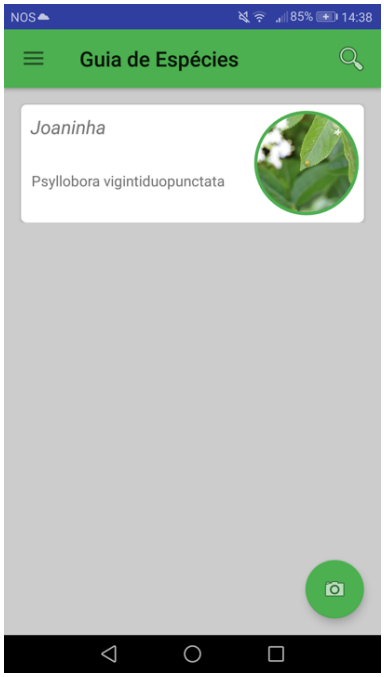

Figure 68 - Species guide activity

As seen, each specie is inside a container that includes the vulgar name, scientific name and photo. The vulgar name is the most known name of that insect, so it makes sense for it to appear first and in a bigger size. The scientific name is still here, as it helps differentiate between common names, for example, “Joaninha” can have multiple scientific names for the same vulgar one. The image helps the user to also identify the specie that it’s looking for more easily.

The species activity is the main activity of this platform: it shows all the details regarding a specie, and all the captures taken. It also shows the position where it can be found. This activity works like a central hub for anything regarding a specie (Figure 69).

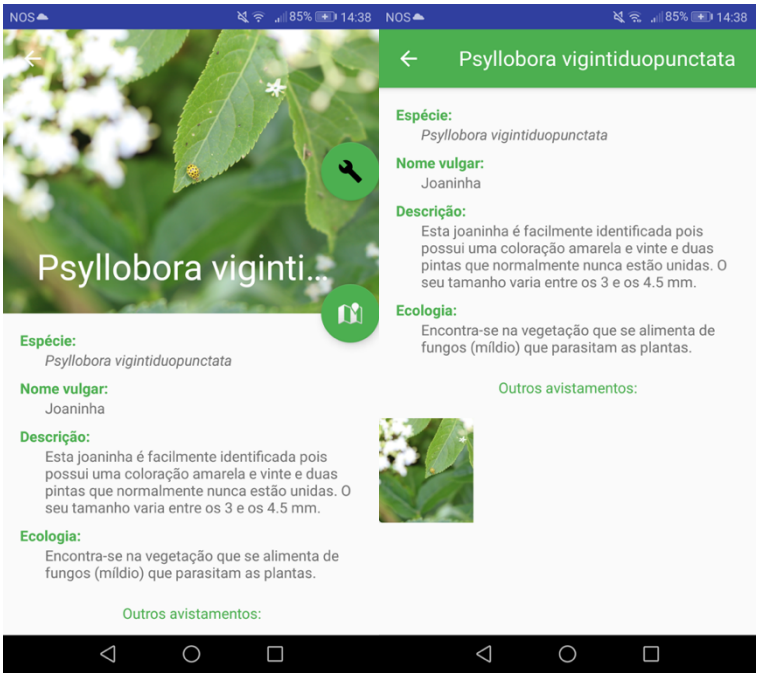

Figure 69 - Specie's information activity

The information shown (Figure 69) is the following: scientific name, vulgar name, ecology and overall description. At the bottom of all the text, there's a grid containing all the photos of that specie. All images are clickable, and every time someone clicks on an image it redirects the user to the page of that capture.

Lastly, the map activity shows all the species distributed in the map (Figure 70). It's possible for the user to search for a specie, and while writing the specie's name the map automatically pans out to the location of the specie. By clicking on the marker, the name of the specie appears and by clicking again, it redirects the user to the specie's hub activity.

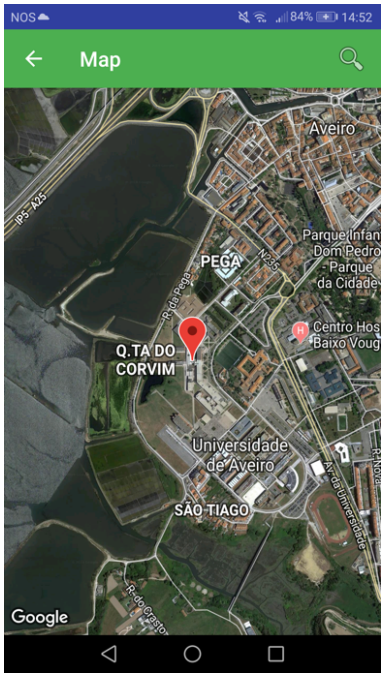

Figure 70 - Species location map activity

Design choices – Web Application

The design choices made with the mobile application were, in a way, transferred to the web application.

The blog page was made using the basic template designed by WordPress (Figure 71). No much change was done to it, as the current design fits the needs proposed initially for this platform.

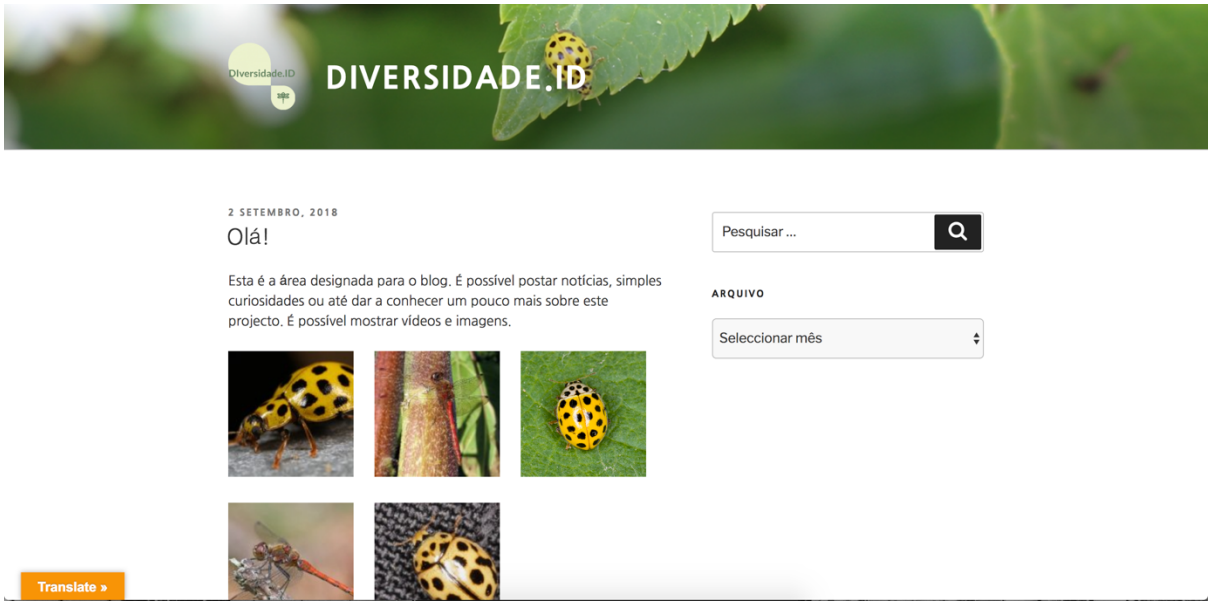

Figure 71 - Blog page

The “List of Species” page lists all the species documented, by alphabetical order (Figure 72). At first, the more species are documented, the bigger this list will get, and it can create a “infinite “page scrolling problem. This can be solved by creating a search function, something that wasn’t designed at the time of the closing of this document. By being ordered alphabetically, it mitigates to a less extent this problem.

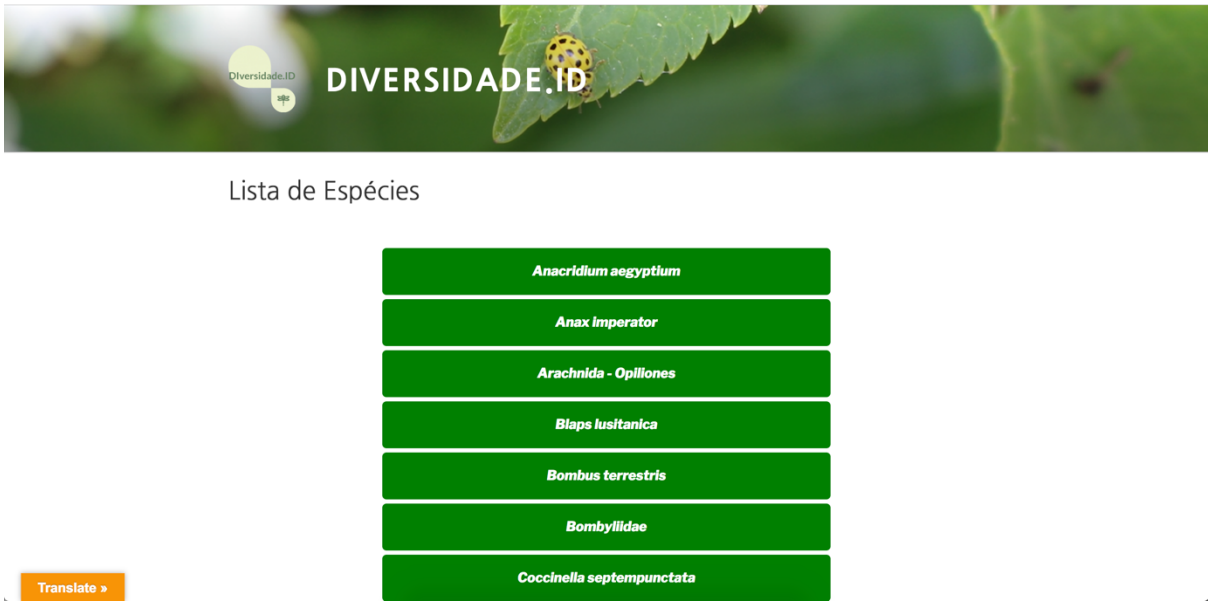

Figure 72 - List of species page

The “Specie Information” page initially was to be designed like the mobile application. This would create a scrolling problem that it can become more problematic as more captures are added, since the web application uses high quality images (compared to the placeholders used in the mobile application). With that in mind, the best way to mitigate this problem was to create a cycling gallery, where the user can control by using the two buttons “Previous” and “Next” (Figure 73). This doesn’t load the page with too much information and keeps it all in the same place.

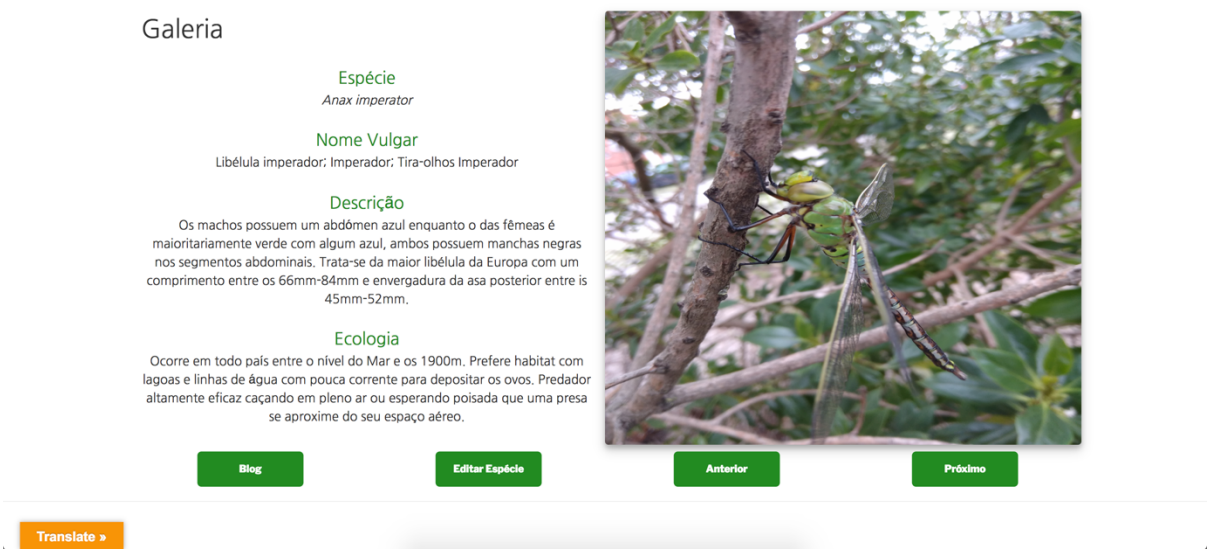

Figure 73 - Specie information page

This page also has the "Blog" button, that redirects the user to entries that contain the specie's name in the blog (Figure 73).

The "Map" page is a simple design – a map container, with the ability to search for the species (Figure 74).

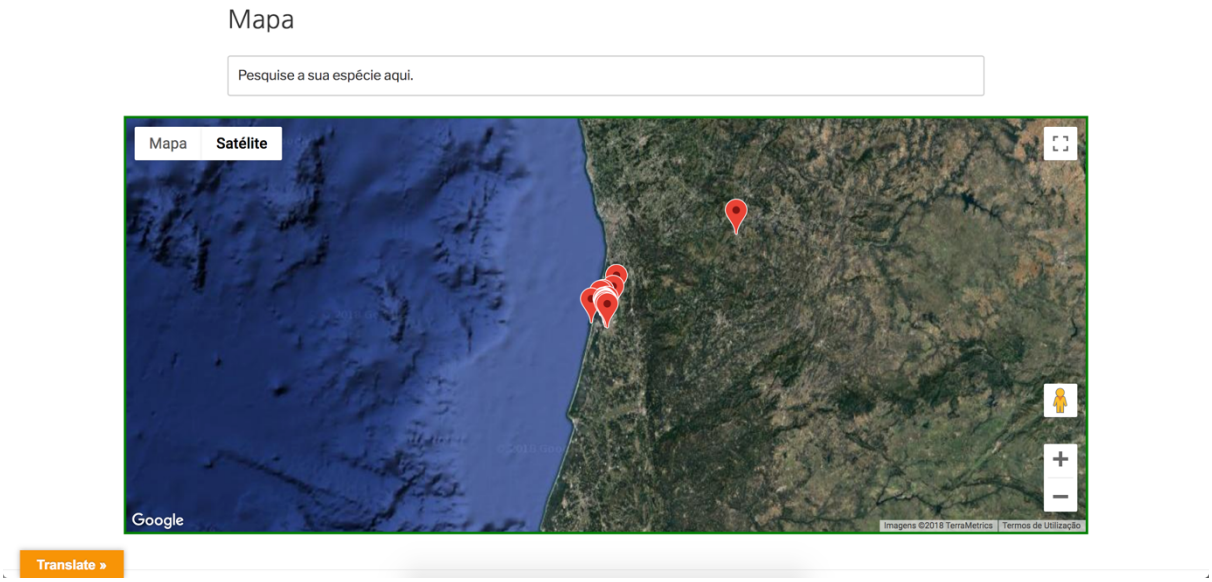

Figure 74 - Map page

The "Pending Captures" page (Figure 75), despite what was previous said about the "infinite" scrolling problem, was design as a scrolling page. This lets the collaborator check and choose which capture it would like to document. The images are still of higher quality, but hiding the captures in a cycling gallery, like we did in the "Specie Information" page, would make the workflow for the collaborator slower.

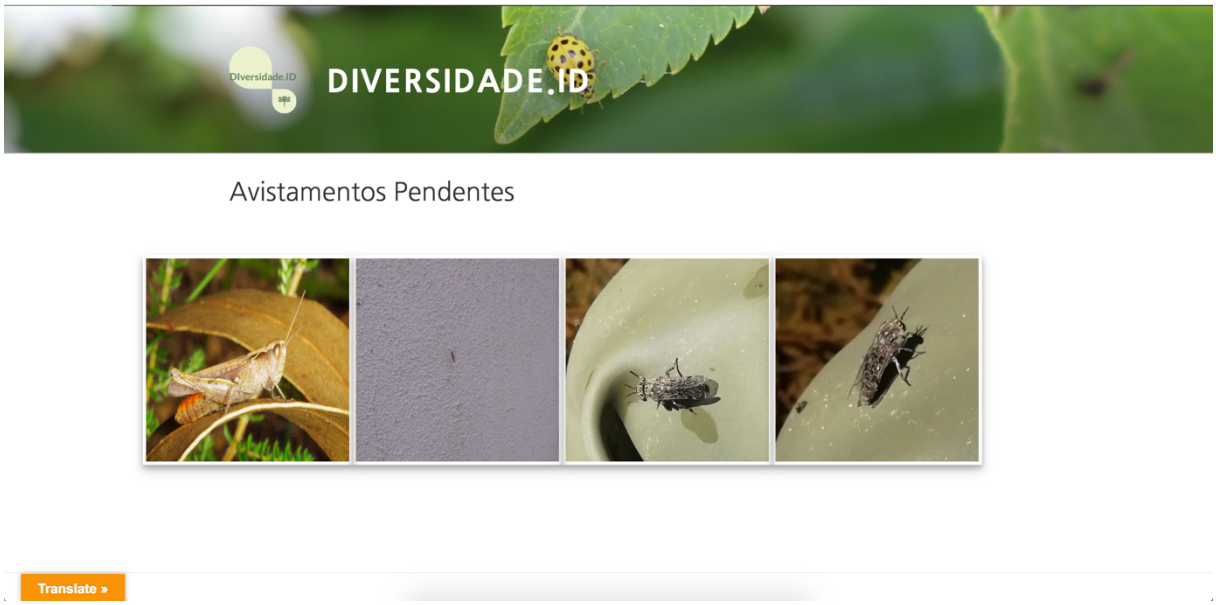

Figure 75 - Pending captures page

Usability testing

The usability tests were made in two phases: the initial one to study the placement of the components, checking the accessibility. The second one was an overall usability testing, with the focus on trying the application on people with no application knowledge. This would serve to test the final design of the mobile application. These tests were designed for the mobile application only.

The testers were given a number of actions to do, with the flow expected from a citizen user. They would test the two type of users, citizen and collaborator. The following list shows the directions:

- Citizen Test:
    1. Make an account
    2. Take a capture
    3. Check information regarding new capture
    4. Check user's captures
    5. Check other users captures
    6. Check information from a capture
    7. Check the location of the capture
    8. Check that capture's specie information
    9. Go to Species List
    10. Search for species in the Species List
    11. Check species location map
    12. Search for a specie in the species location map
    13. End session

- Collaborator Test:

1. Make an account
2. Login with the account
3. Take a capture
4. Accept/Reject newly taken capture
5. Edit Information regarding that capture
6. Check user's captures
7. Check other users captures
8. Check information from a capture
9. Check the location of the capture
10. Check that capture's specie information
11. Go to Species List
12. Search for species in the Species List
13. Check species location map
14. Search for a specie in the species location map

The users would need to follow these steps and register the time they would take to finish each step. This time would show how fast the users could recognize an action in the application, giving us a glimpse of how well the design could show the user what he/she wanted to do. If the user took little time to finish an action it means the design is enough self-explanatory.

The usability tests done during this session helped shape what the applications looks final in its final version. All comments and opinions were taken into consideration.

As previous said, this test was divided into two – first and initial design evaluation, and second one and final evaluation and validation (that was included in the Pilot Use scenario).

Usability testing results – First Phase

For the first test, a group of 6 people, with no previous knowledge of the application were used. After a brief explanation of what the application is, the tests were given to the person for him/her to begin the tasks. This was a one testing session scenario, unlike the second one. Consider the following tables:

Task	Actual time (average) in sec.	Expected time in sec.
1. Make an Account	45 seconds	40 seconds
2. Login to created account	1 seconds	5 seconds
3. Check user's captures	1 second	1 second
4. Check other users captures	2 seconds	4 seconds
5. Check information from a capture	1 seconds	3 seconds
6. Check the location of a capture	6 seconds	5 seconds

7. Check capture's specie information	4 seconds	5 seconds
8. Check specie's location	4 seconds	3 seconds
9. Take a capture	6 seconds	4 seconds
10. Check new capture	3 seconds	4 seconds

Table 5 – First phase test 1 average times

Task	Actual time (average) in sec.	Expected time in sec.
11. Make a collaborator account	27 seconds	40 seconds
12. Login to created account	3 seconds	5 seconds
13. Check user's captures	1 second	1 second
14. Check other users captures	2 seconds	4 seconds
15. Check information from a capture	1 second	3 seconds
16. Check the location of a capture	2 seconds	5 seconds
17. Check capture's specie information.	2 seconds	5 seconds
18. Check specie's location	2 seconds	3 seconds
19. Take a capture	5 seconds	4 seconds
20. Accept new capture	16 seconds	10 seconds
21. Edit information of the documented capture	15 seconds	10 seconds

Table 6 – First phase test 2 average times

For the first table, the results are self-expressive: the values registered don't come too far from the expected values. This means that the application has a simple and self-explanatory designs, even with some problems. The most common problem was the recognition of the icons on the buttons, leaving the user wondering where he should click next. The button position was good, as they could immediately recognize that those buttons were important, but the icons were not self-explanatory enough.

There was one user that went into a bigger explanation of what he felt that was wrong with the application, that followed a brief discussion on it with the other users. Eventually they all agreed on the problems faced when testing and was really important on the final design of the application.

The first problem was when clicking the holder of the capture. Clicking on the overall holder would do nothing, only clicking on the "Map" or "Info" button would do their respective actions. This is not feasible, as most people will click on the image itself to learn more about it. For the final version, the holder was clickable, redirecting the user to the information of that capture.

The second problem was the change of “Sightings Guide” to “Species List”. Calling it a guide, it wasn’t clear that this activity was designed as an information hub for the species. Calling it a list shows exactly that – lists all available species and clicking on one of them redirects the user to the information page.

Third problem is on the same nature of the second one: the name change of “New Sightings” to “Pending Sightings”. Using the word pending shows the user that something is waiting for approval, exactly what this activity was designed to do.

Button placement was also discussed but no final agreement was reached on this matter.

Fourth problem was found when editing a specie’s information. Users agreed that it would be useful to have the prior information when adding a new one. If the information needed some correction, the user would need to know what was written there before doing such correction. By adding the already documented information, it’s easier for the user to correct or add new information to the documented specie. This was something that was inserted into the final design.

Fifth and last problem, putting the “Sightings Gallery” activity as the first one. This would be the best, as it would show the user what the application is about. Initially this was an option, but the final design was different.

Usability testing results – Second Phase

The second test was implemented with the Pilot Use scenario, were users were given 2 days to use the application freely, letting them do whatever tasks they wanted and let the application work naturally without the controlled scenario of the first usability test. But, before using the application freely, they would need to follow the same tasks as the first test (to get the average time spent on each task). This would also be used to test the usability and design of the final application.

While two of the users had no problem with following the usability test steps and had no suggestions to make regarding the design of the application, another one had trouble with some of the steps.

Note that average time to completion of each task was not reported on this second test because, despite two of the users having reported the time they have spent on each task, a third one didn’t, rendering the results not conclusive enough. As the nature of this second test was the evaluation of the final design that was built based on the previous test, the average time spent wasn’t as important as in the first one.

The following table will explain, step by step, the problems faced by the users during the usability test.

Task	Tester 1	Tester 2	Tester 3
1. Make an Account	No problems found	No problems found	No problems found

2. Login to created account	No problems found	No problems found	No problems found
3. Check user's captures	No problems found	No problems found	No problems found
4. Check other users captures	No problems found	No problems found	No problems found
5. Check information from a capture	No problems found	No problems found	No problems found
6. Check the location of a capture	No problems found	No problems found	No problems found
7. Check capture's specie information	No problems found	No problems found	(Goal) The initial goal was to, from a capture information activity, go straight to the respective specie's page. (Solution) This user thought that simply clicking on the specie's name was not enough, so it was suggested underlining the name, like a link.
8. Check specie's location	No problems found	No problems found	No problems found
9. Take a capture	No problems found	No problems found	No problems found
10. Check new capture	No problems found	No problems found	No problems found

Table 7 - Second phase Test 1 results

As seen from the table, no major problems were found during the execution of this first usability test, made for the citizen user. On the task number seven, the third tester found problems with the step to take to get from the current capture's page to its respective specie's page. The solution was to underline the name, making it look like a link (like we mostly see on websites) that redirects to somewhere (Figure 76).

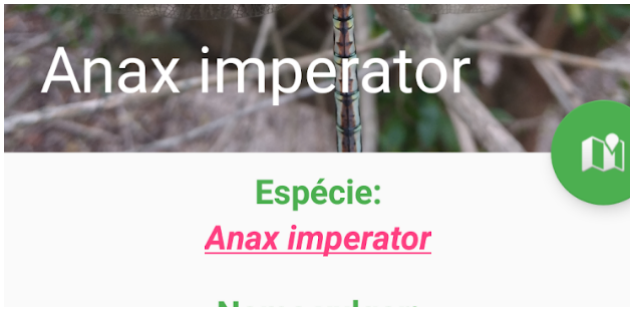

Figure 76 - Feature change based on feedback (Specie's name)

The next table shows the results of the test number two of the usability test done during the Pilot Use scenario.

Task	Tester 1	Tester 2	Tester 3
1. Make a Collaborator Account	No problems found	No problems found	No problems found
2. Login to created account	No problems found	No problems found	No problems found
3. Check user's captures	No problems found	No problems found	No problems found
4. Check other users captures	No problems found	No problems found	No problems found
5. Check information from a capture	No problems found	No problems found	No problems found
6. Check the location of a capture	No problems found	No problems found	No problems found
7. Check capture's specie information	No problems found	No problems found	Same as the test one.
8. Check specie's location	No problems found	No problems found	No problems found
9. Take a capture	No problems found	No problems found	No problems found
10. Accept new capture	No problems found	No problems found	No problems found
11. Edit information of the documented capture	No problems found	No problems found	(Goal) Add the name of the specie, to be used as an identifier. (Error) The user tried to add a specie name with the

			special character ".", and errors appeared. (Solution) Verify the input of the user on this auto complete Text View.

Table 8 - Second phase test 2 results

Once again, mostly no user had problems with the use of the application, minus the third user that found an input verification error. This input error happened because of the way Firebase Realtime Database creates branches – it doesn't accept special characters. As already explained, the name of the specie will be the path of the branch created for this specie in particular. Firebase Realtime Database simply doesn't accept any special characters on its paths. The only way to mitigate this problem is through input verification by the user.

6.3 Pilot use scenario

Considering that the current state of the application met the basic requirements initially proposed, it was considered as the next step a pilot use test. This test would put the application in a real-world context, used by actual users interested in the application. This is an important step because it will show how the application acts when it's doing what it was initially proposed to do, outside of controlled environments.

Pilot scenario methodology

The pilot scenario was divided into two phases: first phase was the use of the application freely, with no controlled scenario, for the users to upload and document species with no restrictions; second phase was the SUS questionnaire.

Initially, the latest version of the application was installed into 5 smartphones belonging to the testers. After a brief explanation of the project and what their purpose would be, it was decided that three would be citizen users and two of them would be collaborators. This would grant the application enough data to test its robustness and performance in real life scenarios.

For two days, the amount of data in the platform grew to more than triple of the amount of data before the Pilot Use scenario. This tested the platform behavior with big amounts of data, and how everything acted when dealing with a variety of species and bigger flow of information of when during the first phase of testing.

At the time of closing this document, some tests are still going so not all results can be considered for the final results of the pilot use scenario.

Those questions fall on the already known diagnostic test called System Usability Scale (SUS). 10 questions, each answered using a value from 1 to 5, where 1 is "Strongly Disagree" and 5 is "Strongly Agree". The questions used were the following:

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.

3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

Table 9 - System Usability Scale (SUS) questions

After using the system, the users would be ready do answer these 10 questions.

Results

During this period, no major errors were found, and the platform behaved as expected.

Like already stated, at the writing of this section, three testers already reported their SUS results, making it able to draw conclusions from them. Keep in mind, one user was a citizen user, while the other two were collaborators.

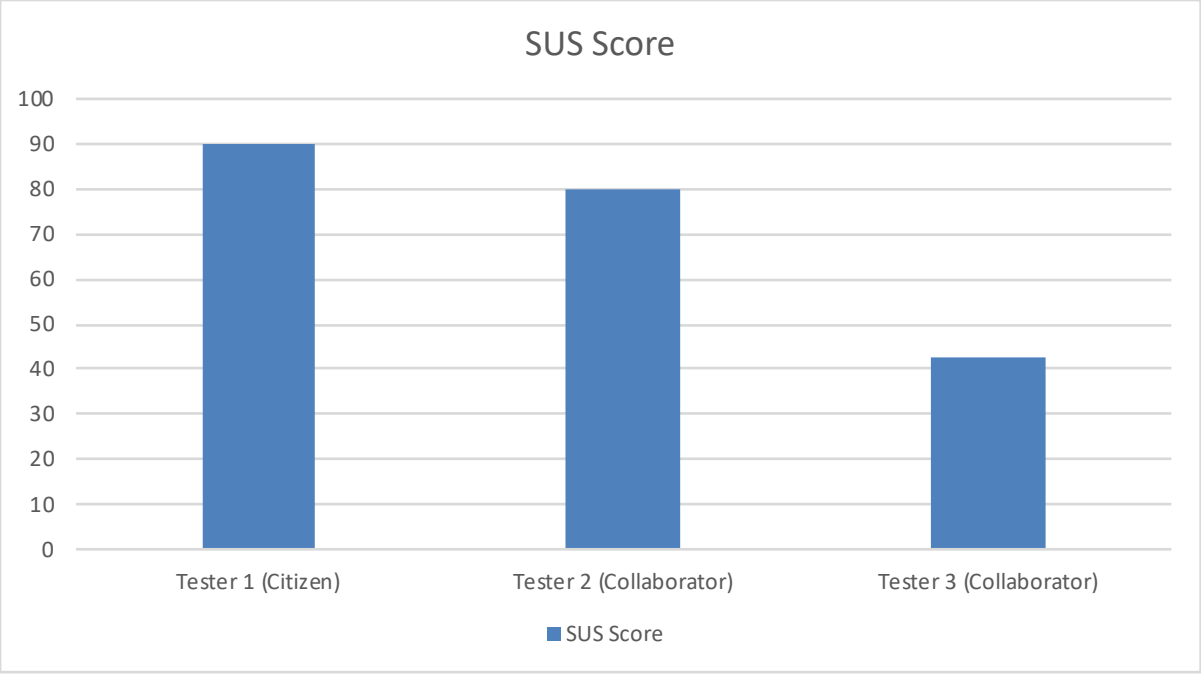

Figure 77 - SUS scores

As seen in the graphic above (Figure 76), two of the users SUS score was very good, while the third one scored below average. The tester 3 was the user that had the most problem with the tasks at hand and the usability of the system.

Another thing that can be taken from the SUS scores is that the overall system complexity, while on a collaborator account, increases and expectedly so. Being a collaborator

brings more options to the application, increasing the actions that can be done on it and, consequently, increasing the complexity of its usage. What was not expect was the low value of the SUS score for the third tester – maybe this can be explained because of the age difference and average application usage between both testers. The actual reasons for this to happen can't be exactly concluded, thanks to the small testers pool used for this pilot use scenario. Clearly, the second tester had no problems while using the application, while the third one struggled to do some more complex actions, such as editing species information.

7 Conclusion

7.1 Work developed

There are already numerous applications that can capture and document species of animals, with a variety of features, but none of them are designed just for insects or with a big focus on entomology like DIdversidade.ID is. They either don't focus on a specified animal type or they offer solutions to different problems other than species location, such as disease control for malaria like MozzWear. diversidade.ID offers a solution to a local problem of documentation of insect species in Baixo Vouga Lagunar. It does that by relying on citizen science, meaning that it uses regular smartphones users to help with the capture and documentation of species. By doing that, it offers a much more extensive pool of resources, speeding up the process of studying and documenting of these species.

diversidade.ID offers a reliable and simple way of using the help of citizens to big areas of study like entomology, while being something of interest to use. All users can make captures, check other users captures, check the location, make them interact with the project itself by capturing the biggest number of species they can while learning about them.

Even though the project is reliable and offers a solution to the problems proposed, during its development it showed some complications. The initial approach was to do our own back-end services, such like webserver with local hosting. Due to costs and because a local server requires constant management, that approach was postponed and switched to another one – use an online synchronized database. That's where Firebase comes in, as it offers a variety of services capable of meeting the needs of this project while mitigating the problems of maintenance and costs. Since the Firebase plan being used currently is the free tier one, it has some limitations to both the Realtime Database and Storage. In the case of this platform growing more, it's possible that this plan tier is not enough and, either the current platform is migrated to a local webserver or the tier plan needs to change to accommodate the amount of data and users it might get.

Another problem is the constant internet connection needed for the application to work. This is an online-only application and considering the conditions where this application can be used, it might have some problems. If the user is on a location with bad connection, the uploads can have problems – on this case, we recommend the user to take pictures offline, store them and eventually upload them when the connection is better.

Even though the memory management problem is mostly mitigated through the use of GLIDE and other algorithm techniques, there might still create problems, depending on the amount of RAM the user's smartphone has. Since this is a heavy image-based application, with information being exchanged all the time, if running in parallel with other memory intensive applications, some problems might appear. During the tests they rarely occurred, but there's a low probability that it might happen. On newer smartphones, this is not actually a problem, but when using the application on lower-end smartphones, the probability of a memory error appear rises – but still, it's very low.

Despite all the problems and errors during development, the application offers a simple solution to the problem initially proposed. This platform was developed with all type of users in mind, keeping the process simple and accessible to anyone interested in it. A smartphone application and website were developed, working as an extension with each other and constantly synchronized.

diversidade.ID was successfully deployed as a platform that lets the user capture and document insect species in the area of Baixo Vouga Lagunar, with the help of entomology experts.

7.2 Evolution and future work

As said before, this platform has some limitations, especially the back-end services. Cost and management requirements were the biggest factors that weighted the decision to change the initial approach of a local back-end webserver and database to cloud hosted ones. So, since the initial approach is still the goal in the long run, the next step would be to make the transition from cloud hosted services to local services. This means transfer the database in Firebase Realtime Database and Firebase Storage into a local one, running on its own machine, and switch the cloud hosting service like DigitalOcean to a local webserver hosted by the university.

Another aspect that should be considered as future work is the development of an iOS application. Not all users have an Android smartphone, so developing an iOS application should be crucial for the platform to reach more users. Even though Android has the biggest market share, as already stated, iOS can't be neglected.

8 References

[1] T. Gura, "Citizen science: Amateur experts," [Online]. Available: https://www.nature.com/nature/journal/v496/n7444/full/nj7444-259a.html.

[2] J. M. Dauer, W. A. Babchuk, W. A. Babchuk, T. Heng-Moss and D. Golick, "In Their Own Words: The Significance of Participant Perceptions in Assessing Entomology Citizen Science Learning Outcomes Using a Mixed Methods Approach," [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5872281/.

[3] J. C. Carlson and M. S. Fox, "Citizen Scientists in Entomology Research - Instant symposiuM," [Online]. Available: https://academic.oup.com/ae/article/58/1/8/1754561.

[4] "Vacaloura," [Online]. Available: http://www.vacaloura.pt/projeto/.

[5] Zooniverse, "Zooniverse," [Online]. Available: https://www.zooniverse.org/about.

[6] "Cornell University Library," [Online]. Available: https://arxiv.org/abs/1711.06346.

[7] "iRecord," [Online]. Available: https://www.brc.ac.uk/irecord/.

[8] "iNaturalist," [Online]. Available: https://www.inaturalist.org/.

[9] "Mammal," [Online]. Available: http://www.mammal.org.uk/volunteering/mammal-mapper/.

[10] "NatureSpot," [Online]. Available: https://www.naturespot.org.uk/.

[11] "Google Mobile Services," [Online]. Available: https://www.android.com/gms/.

[12] "Android Brands," [Online]. Available: https://source.android.com/setup/start/brands.

[13] "Android Licenses," [Online]. Available: https://source.android.com/setup/start/licenses.

[14] "Statista," [Online]. Available: https://www.statista.com/statistics/263453/global-market-share-held-by-smartphone-operating-systems/.

[15] "Statista," [Online]. Available: https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/.

[16] "Android Introduction," [Online]. Available: https://developer.android.com/guide/.

[17] "Google’s iron grip on Android," [Online]. Available: https://arstechnica.com/gadgets/2018/07/googles-iron-grip-on-android-controlling-open-source-by-any-means-necessary/.

[18] "Firebase Realtime Database Documentation," [Online]. Available: https://firebase.google.com/docs/database/.

[19] "Firebase Storage Documentation," [Online]. Available: https://firebase.google.com/docs/storage/.

[20] w3techs, "CMS Market Share," [Online]. Available: https://w3techs.com/technologies/history_overview/content_management/ms/y.

[21] Forbes, "With 60 Million Websites, WordPress Rules The Web. So Where's The Money?," [Online]. Available: https://www.forbes.com/sites/jjcolao/2012/09/05/the-internets-mother-tongue/.

[22] builtWith, "CMS Usage Distribution in the Top 1 Million Sites," [Online]. Available: https://trends.builtwith.com/cms.

[23] WPShout, "WordPress and the Front Controller Design Pattern," [Online]. Available: https://wpshout.com/wordpress-front-controller/.

[24] DB-Engines, "DB-Engines Ranking," [Online]. Available: https://db-engines.com/en/ranking.

[25] Washington State University, "The What & Why Of Entomology," [Online]. Available: http://entomology.wsu.edu/prospective-students/the-what-why-of-entomology/.

[26] "Statista," [Online]. Available: https://www.statista.com/statistics/263453/global-market-share-held-by-smartphone-operating-systems/.

[27] [Online]. Available: https://arstechnica.com/gadgets/2018/07/googles-iron-grip-on-android-controlling-open-source-by-any-means-necessary/.

9 Annexes

Annex 1 – Usability Tests

This was the usability test used on both the first and second phase of the usability testing chapter. They're in Portuguese, as the testing was done using Portuguese users.

Teste: Usabilidade da aplicação móvel (Android)

Autor: João Carlos Costa Abrunhosa

Nº Mecanográfico: 65245

Curso: Mestrado Integrado de Computadores e Telemática

Email: joao.abrunhosa@ua.pt

Passos: Deverão colocar um símbolo "certo" ou "errado" (à frente da tarefa) se a tarefa foi realizada ou não, respetivamente. Fazer uma contagem do tempo que demorou a realizar a tarefa, de forma aproximada e colocar ao lado do símbolo.

Teste 1

1. Fazer uma conta de teste (email/password) - não clicar no check "Colaborador/Investigador"
2. Fazer avistamento (tirar foto)
3. Verificar informações do novo avistamento
4. Ver seus avistamentos
5. Ver avistamentos de outros utilizadores
6. Ver informações de um avistamento
7. Ver localização do avistamento
8. Ver informações da espécie desse avistamento
9. Ver o guia das espécies
10. Pesquisar por espécies no guia

11. Ver mapa das espécies
12. Pesquisar por uma espécie existente no mapa de espécies
13. Terminar sessão

Teste 2

14. Fazer uma conta de teste de investigador (email/password) - clicar no check "Colaborador/Investigador";
15. Fazer login com a conta criada
16. Fazer avistamento (tirar foto)
17. Aceitar/Rejeitar novo avistamento
18. Editar informações desse novo avistamento
19. Ver seus avistamentos
20. Ver avistamentos de outros utilizadores
21. Ver informações de um avistamento
22. Ver localização do avistamento
23. Ver informações da espécie desse avistamento
24. Ver o guia das espécies
25. Pesquisar por espécies no guia
26. Ver mapa das espécies
27. Pesquisar por uma espécie existente no mapa de espécies

Nome do Participante:

Idade:

Annex 2 – System Usability Scale

Questões

Meta um número à frente da questão de 1 (Discordo Fortemente) a 5 (Concordo Fortemente) em termos de usabilidade da aplicação.

1. Acho que iria usar este sistema frequentemente.
2. O sistema é demasiado complexo para utilizar.
3. Achei o sistema fácil de utilizar.
4. Não necessito de uma pessoa técnica para me ajudar a usar esta aplicação.
5. Achei as funções do sistema bem integradas.
6. O sistema tem demasiada inconsistência.
7. Acho que a maior parte das pessoas saberia usar esta aplicação.
8. Senti o sistema estranho de usar.
9. Senti-me confiante ao usar o sistema.
10. É preciso aprender antes de usar a aplicação.

Nome do Participante:

Idade: