Universidad San Jorge

Escuela de Arquitectura y Tecnología

Grado en Ingeniería Informática

Final Project

System for Generation and Management of Digital Guides for Emergencies in Android

Project Author: Bryan del Cristo Pérez Ramírez

Project Director: Jaime Font Burdeus

Zaragoza, September 10th, 2020

Este trabajo constituye parte de mi candidatura para la obtención del título de Graduado en Ingeniería Informática por la Universidad San Jorge y no ha sido entregado previamente (o simultáneamente) para la obtención de cualquier otro título.

Este documento es el resultado de mi propio trabajo, excepto donde de otra manera esté indicado y referido.

Doy mi consentimiento para que se archive este trabajo en la biblioteca universitaria de Universidad San Jorge, donde se puede facilitar su consulta.

Firma **Fecha**

10 de septiembre de 2020

Acknowledgments

To my aunt, who introduced the concept of this project and has promoted it ever since.

To my girlfriend, who is my biggest source of support and without whom this project would not be the same.

To my family and friends, who always encouraged me to keep pushing forward and gave me the opportunity of living this amazing experience.

To the project director, to whom I will be eternally grateful for his continuous advice and guidance. It has been a pleasure to work with such a professional.

To all the professors, whose teachings and assistance have greatly contributed to our overall development.

To the hope for a better society, where we put our differences aside and care more for each other.

Table of Contents

Table of figures

Table of tables

Resumen

Este proyecto nace de la necesidad de convertir una guía sobre la actuación en caso de emergencias, promovida y avalada por el Gobierno de Canarias, del papel al formato móvil, con el fin de hacerla más accesible y sensibilizar a la sociedad sobre la importancia de estar bien informados. Para ello, en este documento que forma parte del Proyecto Fin de Grado para la obtención del título de Ingeniería Informática, se procederá al análisis, diseño e implementación de un sistema con el que no sólo se pueda realizar la conversión de ésta, sino que también permita la inclusión de cualquier otra guía de estas características.

Con este objetivo, se crea una solución compuesta por tres componentes principales: la plataforma web, dónde se crea y gestiona el contenido; la aplicación Android, dónde se visualiza, y el servidor, encargado de la persistencia de los datos y la comunicación entre los componentes. Este sistema permite total flexibilidad al crear guías, con la posibilidad de incluir texto, imágenes, GIFs y vídeos, determinar su privacidad, e incluso personalizar su apariencia en la aplicación móvil. Además, ha sido diseñada de principio a fin teniendo en cuenta al usuario, partiendo de un cuestionario para conocer sus intereses hasta su validación del producto final. La solución propuesta cumple con los objetivos iniciales y manifiesta grandes posibilidades de expansión.

Palabras clave: Guía Digital, Emergencias, Aplicación Móvil, Android, Plataforma Web.

Abstract

This project stems from the need to convert a guide about emergency procedures, promoted and endorsed by the Government of the Canary Islands, from paper to mobile format, in order to make it more accessible and raise our society's awareness about the importance of being well informed. To this end, in this document that is part of the Final Degree Project for obtaining the degree in Computer Engineering, we will proceed to the analysis, design and implementation of a system that will not only allow the conversion of this guide, but also allow the inclusion of any other guide with similar characteristics.

With this objective, we create a solution composed of three main components: the web platform, where content is created and managed; the Android application, where it is visualized; and the server, responsible for the persistence of data and communication between the components. This system allows full flexibility when creating guides, with the ability to include text, images, GIFs and videos, determine their privacy, and even customize their appearance in the mobile app. In addition, it has been designed from beginning to end with the user into account, starting from a questionnaire to learn their interests to their validation of the final product. The proposed solution meets the initial objectives and shows great potential for expansion.

Key words: Digital Guide, Emergencies, Mobile Application, Android, Web Platform.

1. Chapter 1: Introduction

Around the year 2002 in Telde, Gran Canaria, a passionate group of teachers, healthcare workers and other experts, decided to come together and coordinate their resources and knowledge in favor of improving health awareness of the community. As a result, the *Grupo Técnico de Coordinación (GTC)* emerged with the aspiration of promoting a favorable change in their society.

Under the leadership of Nieves Martínez Cía, one of the first projects GTC embarked on was designed to help teachers who had previously raised their concern about not knowing how to attend their students in emergency situations. Thanks to their continuous efforts and in collaboration with professionals from the *Servicio Canario de la Salud, Consejería de Sanidad, Centro del Profesorado de Telde, Consejería de Educación* and *Servicio de Urgencias Canario*, in 2006 an extensive emergency guide, specially focused on educational institutions, was released in paper form.

The guide: *Guía de Emergencias Sanitarias en los Centros Educativos* [1], from now on *GESCE,* included the steps needed to confront the most common emergency situations that happen in our society in a day to day basis, with the focus on the ones that are more frequent in schools all over the world. In this way, chronic health conditions as asthma, epilepsy, diabetes, anaphylactic shock caused by food allergies and first aids indications for the most common emergencies such as Cardiopulmonary Resuscitation (CPR), burns, traumas or sunstrokes were all broadly covered.

The lack of knowledge of the teaching staff about these topics has two main effects in the classrooms that mainly affect those who suffer from chronic health conditions: first, a feeling of insecurity is created around these people due to excessive protection, which makes them feel different from the rest of the group and as a side effect, contributes to their isolation; secondly, they may not receive the correct treatment in an emergency case which is, of course, vital for this kind of situation, especially if the possible after-effects depend greatly on the immediacy and quality of the procedure followed.

For this reason, during the following years, practical workshops were given in those institutions that requested it and, in addition, a copy of the guide was distributed by the *Consejería de Educación del Gobierno de Canarias* to all educational centers of the Autonomous Community of the Canary Islands.

Four years later, in 2010, a new opportunity emerged to revitalize and improve the previous guide by recording an interactive DVD that would contain a multimedia version of all the content the guide already included but in a more visual and appealing way. Once more, with the objective of promoting awareness about this subject and thanks to the continuous effort of many people behind the project, the last version of the guide was published and distributed throughout all the Canary Islands, but this time revamped and accompanied by a CD.

And all of this takes us to the present, the year 2020 where the digital versions impose over any physical copy and where it is estimated that a 70% of the world population will own a mobile phone and a total of 4.5 billion will be an internet user, as stated in this report by *Cisco* [2]. As these trends are already indicating, the previously mentioned guide needs a serious overhaul, that will focus on adapting it to the new technologies and making it available to anyone that needs it. With this idea in mind, the GESCE coordination group met up again to look for possible ways this could be done, and this is when this project comes into play.

The next logical step the guide could take would be for it to be transformed into a fully digital version and in favor of accessibility, it should be present in the most convenient device, the one we are always carrying around: our mobile phone. With this approach, we could check the guide's contents at any given moment thus, easing the access to the information and allowing everyone to learn how to act during an emergency.

Now that we have a clear objective, we could just focus this project in carrying out the conversion to mobile phones but, why not go a step further? What if instead of just adapting one single guide, we build a platform that allows the creation and further management of digital guides for mobile devices?

With this idea in mind, this final degree project will provide the tools needed for this change to happen: a web interface through which new content may be created, modified or deleted, the desired mobile application where all previously created information may be consulted, the persistent storage to ensure data is always available and of course, the necessary communication between all of these elements, which all together will conform our Platform for Generation and Management of Digital Guides for Android.

2. Chapter 2: State of the Art

If we think about it, a guide is just a list of actions that must be followed to achieve a specific goal. In this sense we can find guides that talk about literally any subject: cooking, gaming, education, sports… In some cases, the steps may not be as easy as step one, two and three but in one way or another, there is always a sequence of events that help you get to a final objective.

So, although our engine will allow practically any type of guide to be created, we will study the ones with a sanitary or emergency focus, since the grounds of this project is to achieve the adaptation of *GESCE* to the digital world. Therefore, as a base and inspiration of what can be done, and of course, be improved on, we are going to investigate the different existing guides adapted to mobile phones. We will focus on their interfaces, the structure of the information, what extra options do they have, etc. in short, all their different features with the objective of creating a better version ourselves.

Talking about mobile phones, we are also going to evaluate the different operative systems we can develop our application in, being the two most well-known options Android and iOS, with the objective of finding through which of these platforms we will be able to obtain better results and eventually reach more potential users.

After doing so, we will investigate the different technologies available to develop our full-stack website, with a special emphasis in the CSS frameworks available, which will determine the looks and responsiveness of the resulting web page.

To conclude this section, we will talk about the Domain Specific Language, from now on DSL, which will be the key component for information to be interpreted correctly throughout all our system.

2.1. Mobile guides

To learn about the different features present in real mobile guide applications, we have selected six of the most used ones and analyzed their key characteristics and how common they are compared to the rest. It is also worth mentioning that we have chosen guides with different purposes to get a more general understanding of the options available.

- *URG (G1):* helps in decision-making tasks with the use of interactive algorithms that approximate diagnoses depending on the symptoms. Also provides extensive information about many areas such as Cardiology, Ophthalmology, Neurology, Psychiatry, etc. In addition, some of the most commonly used scales and calculators to write prescriptions are also available. [3]

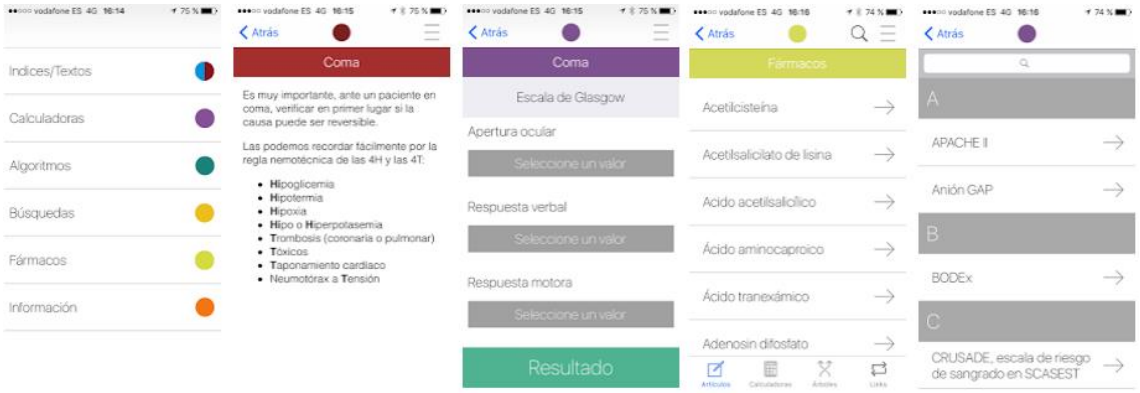


Figure 1 - URG App

Price: Subscription based, 9.99€ per year. Operative systems: Android and iOS.

- *112-SOS Deiak (G2):* allows direct communication with Euskadi Emergency Coordination Centers both through phone call with GPS geolocation or by chat, where you can easily select the type of emergency out of four groups: accident, medical emergency, fire and robbery/assault and you will be further instructed with the steps to follow. [4]

  Price: Free. Operative systems: Android and iOS.

Figure 2 - 112 SOS Deiak App

- *Primeros Auxilios – FICR (G3):* The official International Federation of Red Cross and Red Crescent Societies (IFRC) first aid app gives instant access to the information needed to handle the most common first aid emergencies. With videos, interactive quizzes, and step-by-step tips. It was developed by the Global Disaster Preparedness Center (GDPC) and its contents are adapted to more than 60 countries and available in 40 different languages. [5]

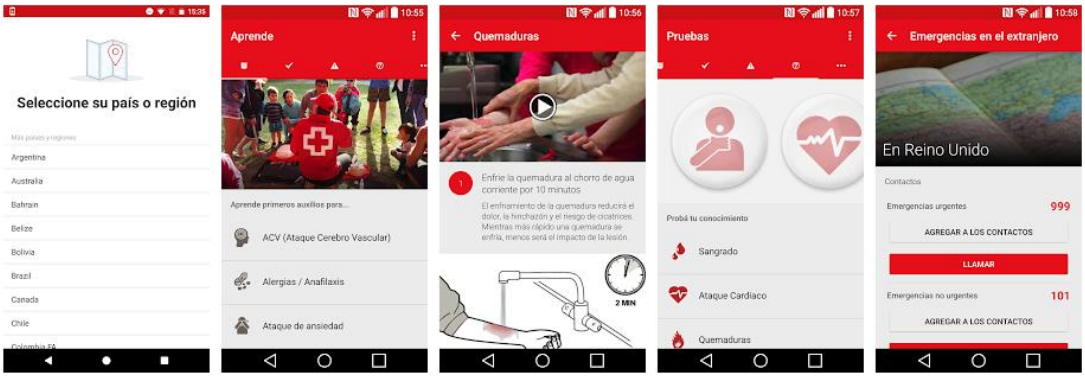


Figure 3 - Primeros Auxilios - FICR App

Price: Free. Operative systems: Android and iOS.

- *Manual de Primeros Auxilios Offline (G4):* Simple step-by-step instructions to guide everyday first aid scenarios. Its main selling point is that all the collected content it available offline. [6]

Figure 4 - Manual de Primeros Auxilios Offline App

Price: Free (with lots of ads). Operative systems: Android.

- *Yo Primeros Auxilios Ribera Salud (G5):* made by the group *Ribera Salud* this app offers basic knowledge about how to act in case of emergency situations, thoroughly reviewed by their own healthcare professionals. [7]

Figure 5 - Yo Primeros Auxilios Ribera Salud App

Price: Free. Operative systems: iOS.

- *Guía Mensa (G6):* digital version of the Spanish *Guía de Terapéutica Antimicrobiana,* annually published since 1990. Contains the most updated information for infectious pathologies, with data about more than 800 different microorganisms. Also includes several scales for choosing the correct antibiotic, its recommended dose and interval of administration.

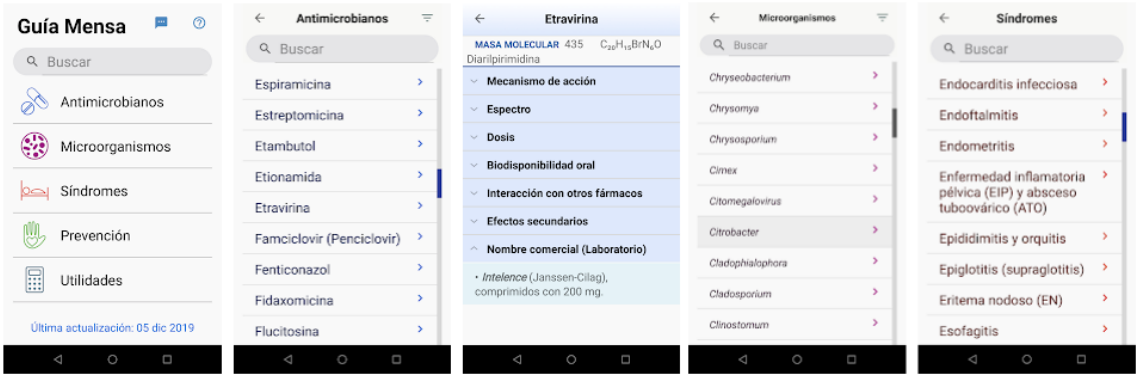


Figure 6 - Guía Mensa App

Price: Subscription based, 19.99€ per year. Operative systems: Android and iOS.

Now that we have some general information about them, we can compare the features each of them includes:

	G1	G2	G3	G4	G5	G6
Action plans for emergency situations	■		■	■	■	
Action plans for first aids			■	■	■	
Action plans with just text	■		■	■	■	
Action plans with text and images…			■	■	■	
… and videos			■		Inoperable	
Disease diagnosis based on symptoms	■					■
Calculate medication dose	■					■
Emergency chat / SMS with location		■			■	
Geolocated call to emergency number		■	■		■	
Other non-sanitary emergencies			■			
Games/questions to test knowledge			■			
Works without internet (after setup)		■	■	■	■	■
Flexible (localization, languages…)			■	■		

Table 1 - Main features in mobile guides

As we can see, these guides offer a wide array of options, some more specific to the app and others are more common throughout. With this information we can learn what can be done and adapt the best solutions to our project; nevertheless, we should especially focus, at least at first, in the most common ones since they will serve as a starting point for our digital guide creator.

Apart from *112 SOS Deiak* and *Guía Mensa,* which have very specific purposes, the rest of the studied apps include some sort of action plan for emergencies or first aid events but although it is very common to find the steps to follow in text format and sometimes even accompanied by some images, videos on the other hand are not that frequent, and on top of this, they only worked as expected in one of the studied apps.

Another feature that seems to be present in most of these apps is the possibility to call an emergency number, an option that is certainly interesting since it can directly provide the location of the caller or even already anticipate the emergency situation that is happening in the case of *112 SOS Deiak*. This seems to be an important option to include, so we note it down for the future.

The last point to mention from this table is that almost all of the apps worked without the need of internet connection, which allows them to be checked at any given moment, even including emergency situations when maybe there is no other help source available (no mobile signal or other people around you for example). Therefore, we should also try to accomplish this result.

2.2. Mobile platforms

We now have some sort of idea about what the market offers in relation to digital guides but something we have not mentioned is what mobile platforms we are going to support. As per stated in this article by *Stat Counter* [8] as per May 2020, there are several mobile Operating Systems in use worldwide but the majority of market share is actually disputed by two contenders: *iOS* and *Android,* with the biggest portion corresponding to this last one, with a difference of about 45%, as shown in the following table and figure.

Android	iOS	KaiOS	Unknown	Samsung	Windows	Series 40	Nokia Unknown	Tizen	Linux	BlackBerry OS	SymbianOS	Other
72,60	26,72	0,2	0,11	0,21	0,03	0,02	0,02	0,02	0,02	0,01	0,01	0,01

Table 2 - Mobile Operating System Market Share Worldwide May 2020

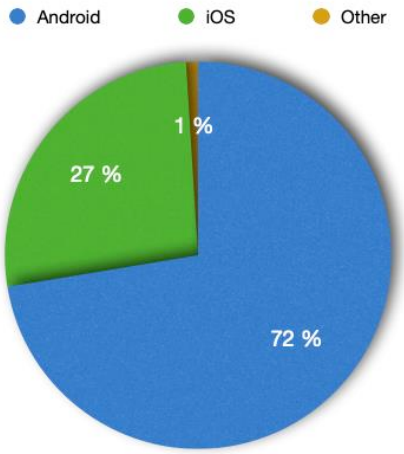


Figure 7 - Mobile operating system market share worldwide approximation

Although the biggest portion corresponds to Android terminals, a considerable percentage belongs to iPhones. If we wanted our app to reach the maximum amount of people, the ideal solution would be to develop it at least in both of these operative systems. Nevertheless, we must take into consideration that our time is limited, as well as our resources, so we should prioritize one option or the other.

If we choose to develop our application in Android, we need to take something else into consideration, and that is the different platform versions available. It is important to choose wisely which API version we are going to develop our app in because it will determine the number of users that will be able to use it. We must mention that although there are many different versions, each new one released is retro compatible, this means that if it works in a specific version, all the following ones will in theory also work.

As we can find in this Android report [9] in a period of seven days before June 1st this is the current Android version distribution:

Version Name	Distribution %
Gingerbread	0,3
Ice Cream Sandwich	0,3
Jelly Bean	3,2
KitKat	6,9
Lollipop	14,5
Marshmallow	16,9
Nougat	19,2
Oreo	28,3
Pie	10,4

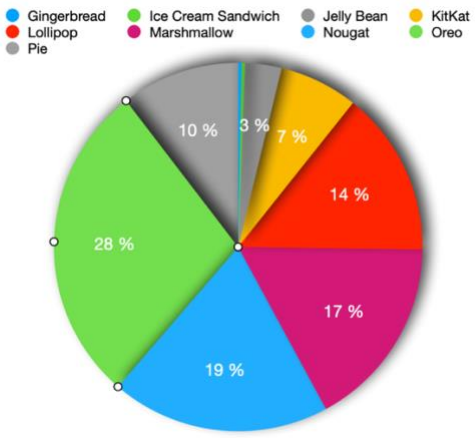


Figure 8 - Android platform versions distribution

The biggest group of devices is running in Oreo but as expected with the current Android market, some previous versions like Nougat, Marshmallow or Lollipop are also used by a considerable number of devices. Therefore, we must choose the option that does not limit us in the development process but that also includes the maximum number of users possible.

In addition, Android has another big problem for developers and this is that on top of there being different versions of the base operative system as we saw in Figure 8, each mobile phone brand that integrates it ends up changing it and customizing it on their own way (those are the benefits of it being open source) but this also means that making an app that works and adapts perfectly to each of the possible versions is nearly impossible.

Meanwhile the iOS version distribution throughout all their terminals is far simpler, especially due to the fact that Apple insists their users much more on keeping their phones updated. In the following figure we can see a report done by the App Store [10] that illustrates the current iOS version usage as per January 27, 2020:

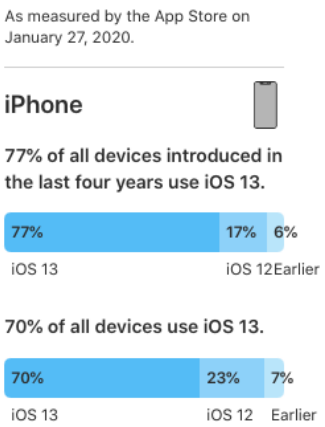


Figure 9 - iOS Version Distribution

Unlike Android, only iPhones run iOS, which makes it far easier to develop an application that adapts to all the different models. Nevertheless, there is a caveat, if we wanted to develop for iOS, we would need to learn their proprietary programming language Swift and learn how everything works from scratch. On the other hand, Android is developed in Java or Kotlin; Java being our preferred choice since it is a language we are much more familiar with and is older than Kotlin, so the documentation we might need will be more exhaustive for Java as well.

From this analysis our conclusion is that as we are more familiarized with Android development and the potential amount of people to reach is also greater, although we might not achieve perfect results in every mobile phone, with our time and resources, we will develop the Android version for this project, leaving the iOS version as a future development.

2.3. Website

So far, we have only talked about the mobile related part of our project but there is a whole other dimension about it: the website. Through it, we will give the users tools to create or adapt an existing guide to the digital format. Therefore, before we start the development, we must learn about the technologies we can make use of to produce the best result possible, always trying to achieve a responsive design to ensure it will be compatible with any user device.

2.3.1. Frontend

Regarding the visible part of the web, we will use the most common language by far and this is JavaScript. This is a language which is fast, adapts really well to the different browsers (specially Google Chrome, the most used) and all in all, is probably the best frontend language to use at the moment. This can be certified by this report [11] by *W3Techs* that states that as per 6th of June of 2020 96% of all the websites use JavaScript. Undoubtedly, we will also make use of both HTML and CSS since they are the base for any web we may imagine.

Coding in pure CSS is possible but obtaining good visual results is quite difficult, since it requires lots of previous experience and have an extensive knowledge of visual design. As we are not experts in this topic, we are going to investigate different frameworks that will do the hard work for us while still achieving really good results. Five of the most used CSS frameworks in the present and the ones we are going to analyze are: *Bootstrap, Foundation, Semantic UI, Pure* and *Ulkit.*

Each of them have their unique characteristics and specific areas where they excel the most. The best part is that all of them are modular which means that you do not need to use all of their components but just the ones you need. Furthermore, it is even possible to mix components of different frameworks at the same time, allowing great flexibility.

To know what each of the options have to offer, let's include the official description of each one of them:

- **Bootstrap**: "*Quickly design and customize responsive mobile-first sites with Bootstrap, the world's most popular front-end open source toolkit, featuring Sass variables and mixins, responsive grid system, extensive prebuilt components, and powerful JavaScript plugins.*" [12]
- **Foundation**: "*A Framework for any device, medium, and accessibility. Foundation is a family of responsive front-end frameworks that make it easy to design beautiful responsive websites, apps and emails that look amazing on any device. Foundation is semantic, readable, flexible, and completely customizable.*" [13]
- **Semantic UI**: "*Semantic is a development framework that helps create beautiful, responsive layouts using human-friendly HTML."* [14]

- **Pure:** "*A set of small, responsive CSS modules that you can use in every web project. Pure is ridiculously tiny. The entire set of modules clocks in at 3.7KB minified and gzipped. Crafted with mobile devices in mind, it was important to us to keep our file sizes small, and every line of CSS was carefully considered.*" [15]
- **Ulkit:** "*A lightweight and modular front-end framework for developing fast and powerful web interfaces.*" [16]

Now that we have seen how each of these frameworks define themselves, let's analyze their differences in a comparison table:

	Bootstrap	**Foundation**	**Semantic UI**	**Pure**	**Ulkit**
Released	2011	2011	2013	2013	2013
Core principles	Responsive Web Design (RWD) and mobile first	RWD, mobile first, semantic	Semantic, tag ambivalence, responsive	Scalable and Modular Architecture for CSS (SMACSS), minimalism	RWD, mobile first
GitHub Popularity	142k Stars	28.7 Stars k	48.1k Stars	20.9k Stars	15.9k Stars
Size	592 KB	233 KB	1.8 MB	3.8 KB	374 KB
Responsive	Yes	Yes	Yes	Yes	Yes
Modular	Yes	Yes	Yes	Yes	Yes
Free templates	Yes	Yes	Yes	Yes	Yes
Documentation	Excellent	Basic	Very good	Good	Good
GitHub repository	Link	Link	Link	Link	Link

Table 3 - CSS frameworks comparison

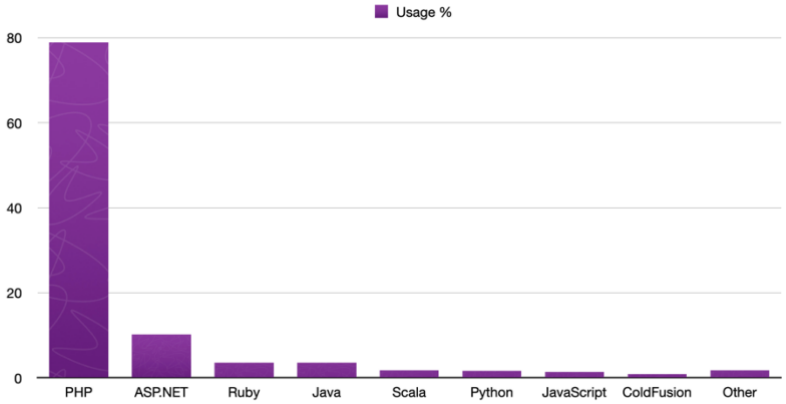
As we can see, most of them are pretty similar in characteristics so we must decide which one to use in relation to how well they will adapt to our project. Our objective is not to create a perfectly designed website, but a functional one that is somewhat visually appealing without the need of investing too much time on design, since it is not the focus of this project. Due to the fact that from the previous mentioned frameworks, Bootstrap is the only one we have some previous experience in, is the most popular one without doubt and has a very well organized and thorough documentation, it looks like the best option to go for, but how does it work?

Using *Bootstrap* is as simple as adding some special attributes to our HTML which provide a completely different look to our website and most importantly, a responsive design that adapts to almost any screen size and characteristics. Its many free templates, community support, flexibility (its open source) and that it works specially well with the frontend languages we have decided to use, will surely help in easing our website development.

2.3.2. Backend

In the case of server-side languages there are many options we could use, each one of them with their unique characteristics and focus. Nevertheless, there is one option that stands out from the rest, as stated by this *W3Techs* report [17] which we summarize in the next figure:

Language	Usage %
PHP	78,9
ASP.NET	10,2
Ruby	3,5
Java	3,5
Scala	1,7
Python	1,6
JavaScript	1,4
ColdFusion	0,9
Other	1,8



Something to take into consideration is that websites may use more than one of these languages at the same time but nevertheless it is clear that PHP is the main option considered by many. The key reasons behind this choice are fairly simple: it is free, it adapts to many databases languages and includes lots of well thought functions that make server-side scripting that much simpler. Combine this with its ability to run on Windows, Linux and Unix servers, its great online documentation and community, and as a result, we obtain the most used backend language in the world. If to all of this we add that it is the server-side language we have the most experience in, it makes it the clear choice to use in this project.

3. Chapter 3: Objectives

The objectives we established in the project proposal were the following:

- Study of conventional emergency guides.
- Adaptation of existing guide to the digital format.
- Backend development that allows the mobile app customization.

Though we had a slight idea of the approach we were going to take, during the first phase of the project where we thought in more detail how we were going to address the different components and in general the result we wanted to achieve, it was obvious that our objectives fell short and needed to be extended.

So now that we have a clearer vision of what we are going to do, lets redefine our objectives list:

- Study of conventional emergency guides adapted to mobile phones.

  Adaptation of existing guide to the digital format implies the development of:

  - Web platform for creation and management of guides.
  - Persistent storage system to preserve the information.
  - Mobile application that is capable of showing the created guides.
  - Communication between the different elements.
- Allow through web platform the ability to customize mobile app.

Although it is not a requirement, since the idea of this project came from the possibility to translate GESCE in to the mobile format, it would be interesting for the future of the project that this could be achieved with the resulting version of this work.

4. Chapter 4: Methodology

4.1. Project Analysis

In software engineering, the methodology is the framework used to structure, plan and control the developing process of a product and over time, several different approaches have emerged, each of them with their unique focus and of course, with their own pros and cons. In order to make a wise decision about which one to use, there are several aspects of our project that we must study and will help us determine which options suits as best. In this way, we must take into account:

- Number of people involved
- Budget
- Customer involvement
- Flexibility of timeline
- Project focus and magnitude
- Project complexity
- Project scalability
- Resistance to change

With these characteristics in mind let's analyze them in relation to this project. First and foremost, this is a final degree project which already implies two things: it will be developed by only one person, so processes like daily meetings, team communication or anything actually that has to do with teams may be discarded, and secondly, there is no economic budget to which we must adhere. Nonetheless, we will carry out an economic study to analyze the real-life cost equivalent of this work in a future section.

In relation to the client figure, on one side, we have the people who created GESCE, that do not care as much about the project itself, as long as they get the result they are looking for: the conversion of GESCE to a mobile app. We will communicate with this group from time to time to show them new versions of the work done and for testing purposes when possible, but they will not actively participate in the decision making done throughout the project. As we must make up for this lack of participation, the more conventional role of a client will be assumed by the project tutor. Thanks to his high availability, we will be able to establish periodic meetings throughout the development that will help in the continuous assessment of the progress being made and on the decision of the next steps to follow in each iteration.

In addition, this project will be developed throughout the school year meaning we will not be able to have a strict working schedule but just work on it whenever we have the time to do so. As a consequence, in the project planification instead of considering real weeks, we will just consider working hours thus, being more adaptable to our situation.

And talking about hours, this final degree project corresponds to twelve ECTS credits where one credit is equivalent to around twenty-five to thirty hours [18], so we can estimate the scope of this project to approximately three hundred hours. In addition, the total available time is limited since we have a set deadline. Therefore, although we may be flexible with our working schedules, we must not forget that we do have a final date to meet.

As for the project itself, it has a certain level of complexity since our architecture is based on two very different pillars: a full-fledged website connected to a database and a customizable Android application that adapts to the contents inserted through the web. On top of this, we do not have a wide experience in these technologies, which will certainly influence the total time we will need to develop the project. In the process, we will also try to create a solid foundation to offer scalability since this product will depend very much in user feedback and therefore it may vary from user demand, so we want to be able to adapt without the need of complex processes or in the worst case, the restructuring of the whole project.

4.2. Methodology choice and adaptation

With this analysis it is clear that we need a methodology that can adapt to frequent changes, is flexible, relies on constant communication with the client by making them part of the process… which all in all indicates us that we will be better off with an Agile methodology. Inside of this category we feel like the two options that could work the best for us would be Extreme Programming (also known as XP) or Scrum.

In theory they are quite similar, but there are some slight differences that can make us opt for one or the other and which we will summarize in the following table (Sources: [19], [20]):

	SCRUM	XP
Iteration length	2 - 4 weeks	1 - 2 weeks
Allow changes in already started iteration	No	Yes, but estimated time must remain the same
Tasks priority	Client suggests but team makes final decision	Client decides
Engineering practices	Not mandatory	Test-driven development, pair programming, simple design, refactoring, etc. are compulsory

Table 4 - SCRUM vs. XP

After this analysis it is decided that XP would work better for our specific case scenario, so we will adopt most of its procedures but also modify the ones that do not apply to our context. XP is based on five values: communication, simplicity, feedback, courage and respect, so let's see what each one of them is about and how we are going to apply them:

- **Communication:** software development depends on adequate communication between the team members, always with the objective of improving and generating better solutions.

  In this case, we do not have other team members to work with, but we do have a client with which we must follow the same standards.

- **Simplicity:** only do the absolutely necessary, keep the system design simple so that it is easier to maintain, support and revise. Only address the requirements you know about.

  This project has an architecture of some complexity, so we will try to create simple deliverables that work and that iteration after iteration start fitting together and afterwards may be improved upon. As the product grows bigger, we will discover new requirements and possibilities to make our software better, which must be documented in *User Stories*, a very useful tool we will cover later in this section.

- **Feedback:** helps teams to identify improvement areas and practices that may be revised. Feedback also supports simplicity.

  This is a product that will improve greatly from feedback, so we will always try to have some functional software so that the client can more easily visualize the work that has been done. We will also rely on acceptance tests, written before the start of each iteration, to demonstrate that the software does what it is supposed to.

- **Courage:** courage is needed to confront difficulties in development like assessing organizational issues that are reducing the team's, stop doing things that do not work and try new approaches, and of course, accept and adapt to received feedback, even when it's tough to do so.

  As this project is done by a single person there will be no support or feedback from other team members, but we still have to keep this relationship and values with the client. In addition, to test development uncertainties we will make use of *spikes* through which we will be able to reduce risks.

As we briefly mentioned before, there are certain tools/practices that XP provides that we can make use of to improve our work flow. The ones we will apply in this project are:

- **User Stories**: short descriptions of what users want from the product. They are mainly used for planning and as reminders of the work that needs to be done. To write these stories we will use the following template:

ID					
TITLE					
DESCRIPTION					
PRIORITY		**RISK**		**ESTIMATE**	

Table 5 - User Stories template

  - **ID:** unique story identifier which will consist of the letters *US* followed by a number, for example: US1. In case this story needs to be expanded or complemented, we will identify it by the previous ID followed by a dot and a number, for example: US1.1.
  - **TITLE:** summarizes what the story is about in a few words.
  - **DESCRIPTION:** detailed explanation of what the client wants. It must be written in colloquial language so that it is understandable by both developers and client and must avoid any type of ambiguity.
  - **PRIORITY:** defined by the client to determine what features they consider as more relevant for the project. It may be marked as *LOW, MEDIUM OR HIGH.*
  - **RISK:** represents the probability that this story will fail or carry out problems. As in priority, it may be marked as *LOW, MEDIUM OR HIGH.*
  - **ESTIMATE:** evaluation of hours it will take to develop this story. In more common XP applications this is done with other metrics like story points, since it is difficult to calculate the time a team will take to produce something, but as this project will be done by one person with a variable schedule this approximation will be simply done in working hours.

- **Spikes:** when estimations are difficult to establish, or developers do not have a clear solution to go for, it is more advisable to dedicate some time to learn more about the issues or explore possible solutions that could work. This will be constantly done through the project since even from the starting point, we are not sure what is going to work and what not, so we will make use of spikes as much as possible to clarify these potential situations.

- **Acceptance tests:** in order to validate that the work done in each iteration does what it is supposed to, before the start of each iteration, we will write a series of tests that will serve as verification. The format to follow will be: present the initial situation, describe a specific event to execute and the expected result. An example of an acceptance test could be: After creating a guide in the website, the user will be able to see it in the Android application.
- **Refactor:** always that it is necessary we will consider refactoring the code to make it as simple and concise as possible and avoid code duplication. With this we will generate a more flexible and reusable software where it will be easier to accommodate new requirements or changes.

4.3. Other tools

To keep control of the progress done and make a better use of the methodology overall, we will rely on these tools:

- **Trello:** to keep track of the tasks of each iteration we use Trello. This tool offers a white board where we can organize our work and have a more visual representation of the state of each iteration. From one glance we can see what needs to be done, what is currently being done and what tasks have already been finished. Boards may be enhanced by *Power-Ups,* as they are known in Trello*,* which basically are small plugins that allow you to do more than one it is offered by default. In this case we use *Scaled by Screenful* which allow us to assign tasks a priority that range from *Very high* to *Very low*, a size, measured in hours, and mark possible dependencies with other tasks.

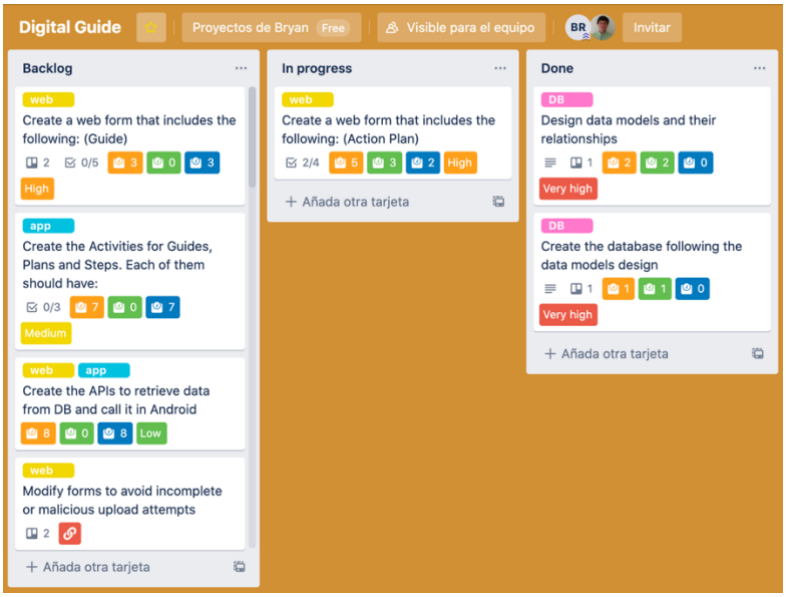


Figure 10 - Trello board example

As a possible alternative, we also invested some time trying other tools like *Jira* by Atlassian. This software is a more professional choice, but certainly more complex too, including many functionalities designed for teams or product managers, that do not apply to our project at all. In favor of simplicity, we discarded Jira and continued using Trello.

- **Git:** As a version control system tool we use Git, which allows us to monitor the different modifications done to our software over time and check them when needed. As a visual interface for Git we use *SourceTree,* through which we control the public repositories we have created on *Github,* one for the website and another one for the mobile app. You may find them in:
    - Web: https://github.com/Bryantheboss/DigitalGuide-Web
    - Android: https://github.com/Bryantheboss/DigitalGuide-Android
- **Excel:** we use a Microsoft Excel sheet to keep track of the hours spent working on the project and as the same times servers as a log of the tasks done.

	A	B	C	D	E	F	G
1		LOG TFG INFORMÁTICA					
2							
3		DATE	TASK	TIME m	TOTAL h	TOTAL worked	TOTAL meetings
4	ITERATION 0: ANALYSIS	9/11/19	Reading manual	60	22,25	20,75	1,5
5		12/11/19	Establishing schedule for possible working hours throughout week	25			
6		13/11/19	Reading about agile methodologies	40			
7			Figuring out general architecture of the project	60			
8			starting project, creating git, everything under version control	60			
9		18/12/19	create, design and finish survey	90			
10			Study State of Art	120			
11		19/12/19	Study State of Art	120			
12			Developing a requirements list	30			
13			Survey	30			
14			MEETING 1	90			
15		21/12/19	Survey	60			
16		9/2/20	Setting up trello and User Stories	60			
17			General design	150			
18		10/2/20	Designing first web form	35			
19			Proof of concept (php)	40			
20		12/2/20	Poc php, upload basic info to BD	70			
21		14/2/20	Updating trello with newer more detailed tasks	45			
22			Poc, php API for Android and display it in app	150			

Figure 11 - Excel log sheet

- **Microsoft Teams, Skype and TeamViewer:** at the time of development of this project, we as a society, experienced a unique situation: the state of alarm in the entire country and several months of quarantine. For this reason, all physical meetings were replaced with their online versions thanks to the use of these tools. In addition, with TeamViewer, an app through which we are able to control another computer remotely, we conducted some small tests with the objective of improving the user experience of our system.

5. Chapter 5: Implementation

5.1. Iteration 0: Analysis

This iteration is designed to understand the problem, how we want to solve it and based on this information, organize the rest of the iterations. We may start by developing a requirements/features list from the requests of three different sources: GTC, A Google Form sent to potential users and ourselves.

The first step we took was to design the survey through the *Google Forms* tool but before that we had to think about what information we wanted to obtain from it. At first, the questions were focused in learning if the respondents knew how to react in very specific emergency scenarios like: *"In case of hyperglycemia, would you know how to act?"* or *"Do you know "Stesolid" or "Buccolam" as remedies to an epileptic crisis?"*. We soon noticed that these questions were too detailed, and we would not gain much from them. For this reason, we decided to rethink our questionnaire and created three groups of questions to obtain the following information: who are our potential clients? How informed are they on the subjects of first aids and emergencies? And what is there opinion in these subjects/what are they interested in learning about.

After some tweaks, we came up with a list of twelve questions, designed to be answered in no more than five minutes, avoiding being too burdensome for the respondents but enough for us to collect the information we were looking for. When ready, we spread it throughout the Universidad San Jorge, GTC, and social media to try to reach the greatest number of respondents as possible. As a result, 201 answers were obtained from the Google Form, resulting in a good mix of user profiles we could take advantage of. Please find the form and the corresponding answer analytics in the Annex section Google Form. PFG Bryan Pérez: Guía digital de emergencias.

From it we learnt that the main age group ranges from 35 to 44 years old and above and most of the respondents were teachers or at least work in an education center, making them our main target. About their general interest in the subject, we received answers of all types, people which were generally not interested at all to people that do care and even like to attend courses to stay updated. Regarding the emergency situations users had already experienced and the subjects they would like to learn about, interestingly enough, everyone had witnessed at least one or more of the emergency situations listed and we received 65 answers of other subjects they were interested in. With the last question of the form, we just wanted to learn if the public thought it would be useful to have a mobile app that gathered all the contents we had previously talked about and from the 201 answers we received, which included students, teachers, other workers of educational institutions and more, 97.5% of them agreed that it was a convenient solution.

Figure 12 - Google form result for mobile guide usefulness

With this, we are ready to gather and analyze all of the requests we received from the different sources. As a result, we obtained the following list:

GTC wants:

- An app that is able to:
  - Show descriptions of first aids or emergency procedures that are straight to the point, with the option of more detailed information if needed.
  - Adapt all the already created multimedia (images and videos) used in GESCE and include them in the previously mentioned procedures.
  - Have the option to call emergency number on button click.

Form users want to:

- Learn about the most common emergency situations.
- Learn about other type of situations: natural disasters, car accidents, drownings, etc.
- An app that includes all the previously mentioned, in an easy and understandable format.

To fulfill the previous objectives, we develop our own list of features we want to achieve:

- Develop a website through which we are able to create guides and action plans.
  - Users must have an account.
  - Must include the option to insert images, GIFs and videos.
- A database that is able to store all the information.
- An app able to adapt the content created in the website and viewed in the mobile format.
  - Should be customizable.
- Create a validation system where trusted sources can verify if the content is correct.

With this information we have a better idea of the scope of the project and we are ready for a new meeting with the client, where we will agree and describe in greater detail the tasks we want to develop. To do this, we write our first User Stories, which are not written in technical language, so that there are no misunderstandings between the client and the developer. These stories are not fixed and as we advance in the development, we will write new ones and expand upon the existing ones, making sure we cover the latest requirements or change their focus if needed.

As a result, the following User Stories are written:

- US1 – Create Action Plans for emergencies through web form (only text).
- US2 – Create Guides through web form (collection of Action Plans).
- US3 – Store Guides and Action Plans in persistent storage.
- US4 – Show Guides with their Action Plans in mobile application.
- US5 – Improve Action Plan creation to allow steps with images and videos.
- US6 – Improve Guide creation to allow personalization.
- US7 – Improve Action Plan creation to allow other media types, customize their layout in Android and add extra options.
- US8 – Login / Authentication system.
- US9 – Feedback / Verification system.

In favor of organization, please find the full description of these and future User Stories in the corresponding Annex section for each Iteration, in this case: Iteration 0 - New User Stories.

5.1.1. Initial planning

This final degree project is equivalent to 12 ECTS credits which translates to approximately 300 hours of work, which have to be distributed between writing this memoir and the development of the project itself. With the information we have right now and with the certainty that the project is going to grow considerably, getting more complicated as we advance, we estimate that we will need about 180 hours for the development, which leaves us with 120 hours for the memoir.

In order to match these time stipulations, we have to determine the amount of XP iterations we want to divide the development in and how long they will last. As a note, we are not counting this Iteration 0 as an ordinary iteration since it serves as the necessary prior analysis before the beginning of the actual work. Having this said, iterations usually go from thirty to forty hours in length, so we made different calculations and found that the most convenient solution would be to have six iterations, each of them averaged in thirty hours of work, making exactly 180 hours. The project memoir on the other side, will be developed in parallel to these iterations, so throughout the development we will also dedicate some hours to its writing, progressively covering all the required information until we have finished it and are satisfied with the final result.

5.1.2. Architecture

Before starting the development of any US, we must invest some time in the design of the architecture, which will determine the tools and general setup we will use. Our architecture has three main components: the content Visualizer, in form of an Android application, the content

Creation Platform, which will be a website, and the server. This last one at the same time is composed of two main parts: the database and the actual web server. The database will be used as persistent storage of all the data of our system and the server will be the one in charge of hosting our website. For Server-App communication, we will make use of HTTP connections and Application Program Interfaces, more commonly known as APIs, through which we will be able to transmit and receive any information we desire.

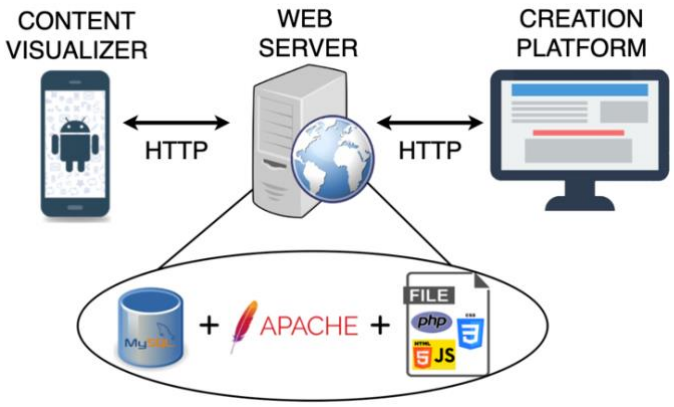


Figure 13 – Architecture components and communication. Sources: [21], [22], [23] [24], [25], [26], [27], [28], [29]

As a side note, we will call the users of the Creation Platform *creators* and the ones who only make use of the Visualizer will be called *consumers*.

Now that we have a clearer vision of how we are going to set up the project, let's compile the tools and technologies that will allow us to carry it out.

- Android Studio IDE version 3.5.3.
  Available at: https://developer.android.com/studio
- Android SDK platform 8.1 (API version 27.3).
- Samsung Galaxy S9.
- XAMPP version 7.2.9: includes the Apache HTTP server, interpreters for PHP and Perl programming languages and from version 5.6.15, a MariaDB database (previously was MySQL but they work in the exact same way).
  Available at: https://www.apachefriends.org/es/download.html
- Google Chrome version 81.0.4044.138.
  Available at: https://www.google.com/intl/es/chrome/
- 000webhost as hosting solution.
  Available at: https://es.000webhost.com/
- Bootstrap CDN version 4.1.3 with jQuery CDN version 3.5.1 and Popper.js CDN version 1.16.0.
  All available at: https://getbootstrap.com/docs/4.1/getting-started/download/
- Sublime Text version 3.2.2 as code editor.
  Available at: https://www.sublimetext.com/3

5.1.3. Proof of Concept

After the initial setup of all the previous tools and the creation of the given repositories needed, we decided we should start by developing a small Proof of Concept, from now on PoC. The idea behind this is to make sure, before starting the real development, that the main concept of the project works and is achievable. Therefore, we create a small and simple version of our architecture and once done, study if it is viable.

First, we design a very simple website which consists of a small form that requests a guide name, emergency name and a description, accompanied by a submit button to upload this information. We then create a new database with just one table and the three corresponding columns.

Figure 14 - PoC - Website and Test table

After this, we write the necessary PHP code to connect to the database and POST the form data. From now on, every time we receive user input from the website, we will follow basic precaution methods to prevent security breaches, especially those related to SQL Injection. This is when SQL statements are inserted through web inputs, normally by the use of scaping characters such as the backslash (\) or SQL specific syntax like semicolon (;), AND or OR. The first rule is to always process user input to handle these characters; this can be done with some built-in PHP functions.

The second measure is to always use prepared SQL statements and bound parameters, which not only prevent SQL Injection altogether but also are more efficient. The main idea behind this is that SQL statements are now templates, where the values are left unspecified using the question mark symbol (?). Afterwards we can bind these values to our parameters, specifying their data type so that the database is able to process them as such. As the statement and parameters are sent separately and even with different protocols, injection can be completely avoided. The improvement in efficiency comes from the fact that the statement only needs to be prepared once, optimizing the process when it will be executed multiple times.

We are now ready to introduce some data through the form and fill our table with some test entries. Then, we create a basic API that returns the data in JSON format.

guideName	emergencyName	description
Guide 1	Emergency 1	Description 1
This is a very useful guide	Cuts	Deep cut.
\\hello//hello	abc//abc\\abc	!"$*$-%QWE242r


```
[
  {
    "guideName": "Guide 1",
    "emergencyName": "Emergency 1",
    "description": "Description 1"
  },
  {
    "guideName": "This is a very useful guide"
    "emergencyName": "Cuts",
    "description": "Deep cut."
  },
  {
    "guideName": "\\\\hello//hello",
    "emergencyName": "abc//abc|\\\\abc",
    "description": "!\"$*$-%QWE242r"
  }
]
```


Figure 15 - PoC - Test table (left) and Basic API response (right)

In Android we just want to display this information in a list and the first step to do so is create an Activity, which basically corresponds to a unique screen with an associated User Interface (UI) known as Layout. In addition, Android Studio provides small working components known as Views which we can use to fill our Layout and create the desired UI. There are many types of Views like TextViews, ImageViews, Buttons, CalendarViews, ProgressBars, etc. but the one we could use right now is called ListView, which in simple words, is just a vertical scrollable list of items. The rows of a ListView are actually Layouts themselves, so Android provides default ones composed of one or two TextViews, since they are simple and can be used in most common cases. Nevertheless, these rows are pretty simple and most of the times we would like to present custom Layouts where the items are arranged in a different way or include other types of Views. Each row is populated with the use of an Adapter, which is in charge of pulling the information from a data source like an array or database and fills the rows with it. Once again, there are default Adapters that can be used but as we will not use the predetermined option, we will have to program the Adapters ourselves.

On top of this there are two ways we can make the use of a ListView more efficient and produce a smoother scrolling of the list: View recycling and the ViewHolder Pattern, that we can apply to our Adapter. The screen of our phones has a fixed size, and this implies that only a set of rows can be shown at a time. For this reason, it is unnecessary to instantiate a View for each item of our Adapter, so when a view is outside of the screen, it can be reused or more commonly known as recycled. On the other hand, the ViewHolder Pattern is used to reduce the number of times we call the function view.findViewById(int), that for those that do not know about Android, is a method that finds the View from a Layout resource file by the use of an identification number. As our items have the same Layout for each of its rows, we should not call this method for each one of them but only once and store this reference in what is called a ViewHolder for it to be reused.

With all of this we are more than ready to create this first version of the app, so we create a new project, a simple class to model the data, make the require call to the API, an Activity with a

ListView and an adapter to populate it with the information we received from the database, all to obtain the following result:

Figure 16 - PoC - Android ListView

5.1.4. Meeting with the client

We spent a total of 31 hours in this analysis phase. The client is satisfied with the work done, especially with the Proof of Concept, where we have verified that the idea we had in mind is achievable: all of our components work independently, are capable of communicating with each other and are able to adapt the information as needed. The foundation is solid and is a good starting point to keep on growing.

Please find this and all future meeting notes in the Annex section 10.9 Meeting Notes.

5.2. Iteration 1

In this iteration we will work in the following User Stories:

- US1 – Create Action Plans for emergencies through web form (only text).
- US2 – Create Guides through web form (collection of Action Plans).
- US3 – Store Guides and Action Plans in persistent storage.
- US4 – Show Guides with their Action Plans in mobile application.

In summary, what we want to achieve is to be able to create simple Guides and Action Plans composed of just text steps, store them in the database and be able to view them in Android.

Now we will divide each of these stories into smaller tasks to be more efficient; we will also write the corresponding tests to verify at the end of the Iteration that we have achieved the desired results. Please find all of this in the Annex section: Iteration 1 - Tasks and Acceptance Tests.

5.2.1. Development

The first logical step to follow is to model our data since defining from the beginning a clear structure of the information will help us in the process of developing the rest of the tasks. Throughout this memoir we have continuously talked about two objects we wanted to model: Guides and Action Plans. A Guide is principally a collection of Action Plans and from our analysis

we can understand that these are just a compilation of Steps to follow, usually in a specific order. From this information we can obtain the following class diagram:

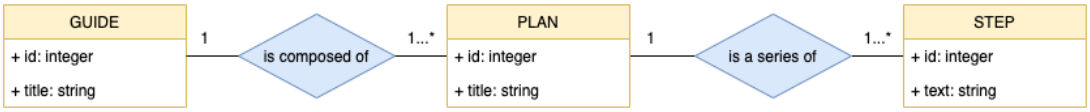


Figure 17 - Initial class diagram

This is the most basic structure we are going to have thus, serving as a starting point. From here we are now able to reproduce it in our database, in which both the classes and the relationships between them are now their own tables:

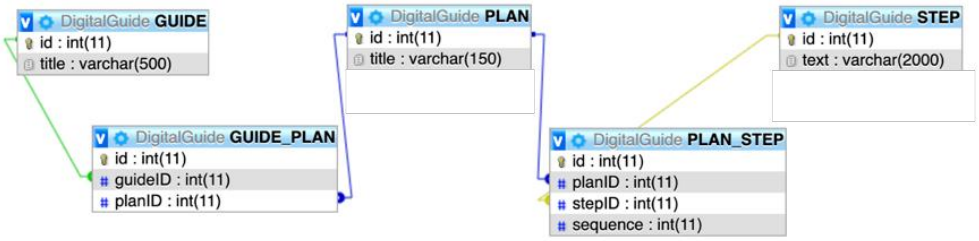


Figure 18 - Initial database structure

The intermediate tables allow flexibility since by holding the references to the classes they relate, we are able to, for example, reuse the same plan in different guides and the same with plans and steps. In this last case, the table PLAN_STEP includes a new field called "sequence" which is used to define their order. With this setup, the labor of the content creators will be simplified, avoiding the unnecessary duplication of data, which also implies a better usage of the database storage.

Now we can improve the website we built for the PoC and we will start with the design of the Action Plan webpage. In this page we want to have a form through which we will obtain a title for the Action Plan and a collection of steps associated to it. We obtained the result seen in Figure 21 where, opposed to the PoC version, it is now developed with Bootstrap, improving the overall aesthetic. On top of this, to make sure we received the requested information, we introduced some frontend validation where all valid inputs will be marked in green whereas the invalid ones will be marked in red, accompanied by a descriptive text, so the user is able to immediately distinguish what is missing when submitting. Here is the result:

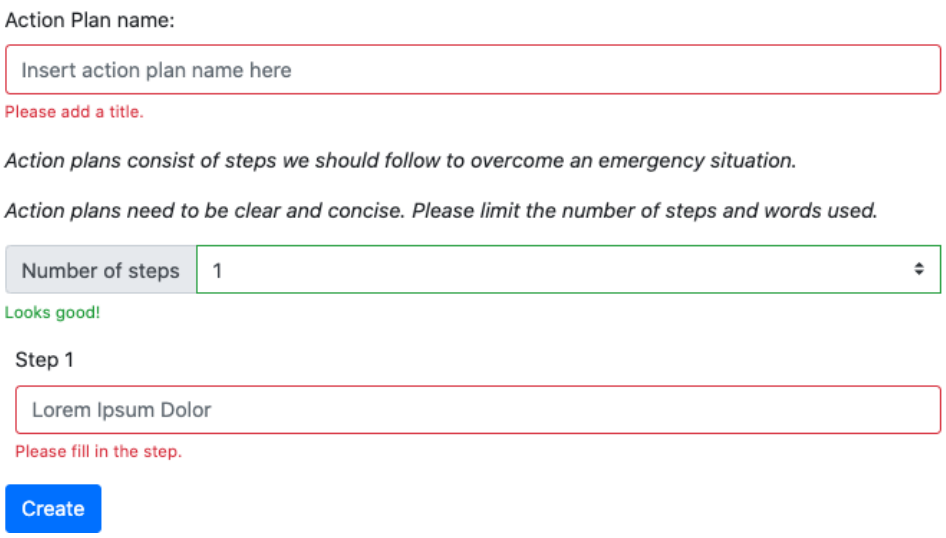


Figure 19 - Frontend validation example

Now that we have confirmed we have received the correct data, we designed the POST functionality to upload this form to the database. As we saw in the database design, Plans and Steps need to be uploaded to separate tables and then create the link between them by storing an entry in the intermediate table. If everything has worked as expected or something has gone wrong in the process, we create dismissible alerts at the top of the page to notify the user:

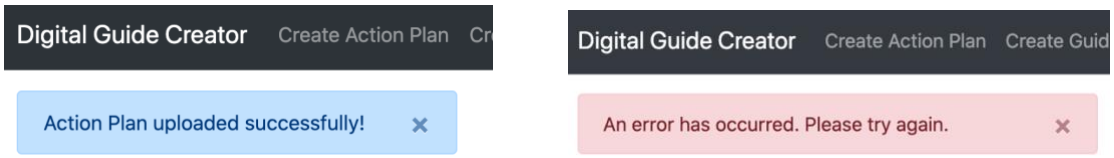


Figure 20 - Dismissible alerts with POST result

After creating some Action Plans, we proceed with the Guide creation webpage which is quite similar to the previous one but of course, asking for other type of data. In this case we just need a title and a way to select from the already created plans.

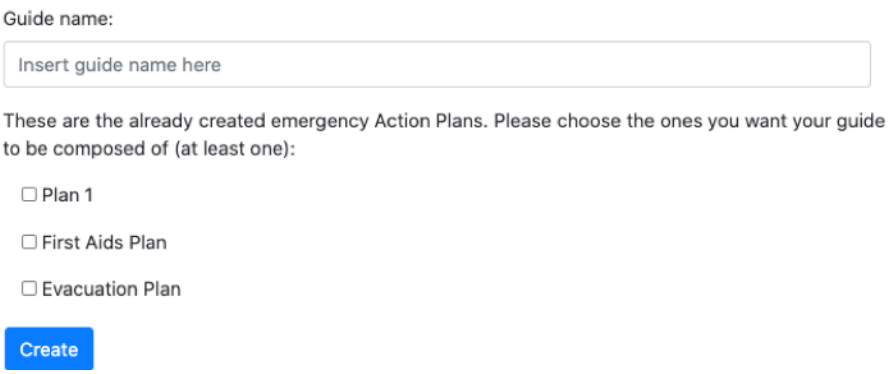


Figure 21 - Create Guide basic form

We then followed the same process as with the other page, including the frontend validation and the POST functionality. As we developed these sites, we were continuously changing back and

forth from the Action Plan to the Guide creation pages, so we decided to include a navigation bar fixed at the top that allowed us to change easily between them.

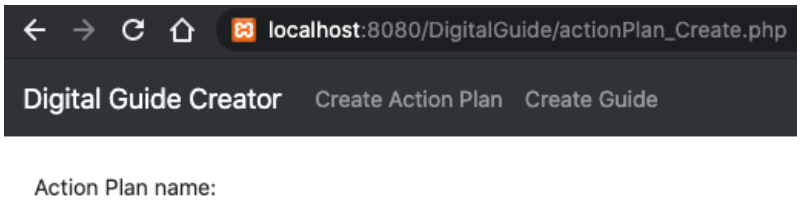


Figure 22 - Initial navigation bar

Now with the Android part, the first step is to design the format of the data we are going to send from the server to the app. We came up with two main options: option A is to send all the information in one unique JSON which is good because you have all the information ready from the get-go but has the major inconvenience that exactly the same Plans could be sent multiple times if more than one guide includes them. Option B consists in separating Guides from Plans, in this way Guides would only include the ids of the plans which are then sent separately. As we can avoid data duplication with the second option, we think it is the better alternative and the one we will use.

Now we just need to code the APIs that return the information in the desired format and when done, we can call the corresponding URLs, which in this case, as we are currently working in our local machine, will look something like: http://localhost:8080/DigitalGuide/API/getGuides.php or getPlans.php, where DigitalGuide is the name of the folder we are storing the project on.

In Android, the first thing we must do is retrieve the data or, at least, the Guides, since they will be the first thing to show. For this reason, we design a simple splash screen that shows a loading animation whilst it fetches for the data in the background. Once it is loaded, we can proceed to display the Guides. Nevertheless, before displaying anything we have to create the models for the new data structures, in this case: Guides, Plans and Steps. As our data will grow in complexity and will be requested throughout all of the application, we decided to implement the Singleton pattern.

This design pattern is based on the restriction of only having one instance of a given class at a time and then provide global access to it so that it can be used from any class in the Android project. In this way, when we receive the data from the API, we will first adapt the response to our models and when ready, we can then insert them into structures like *ArrayLists* to collect them all in one single object. If these lists are found inside our singleton, we can access them from any part of the application very easily: if, as an example, we name our singleton Data and the ArrayList: *guideList*, we would just need to write *Data.getInstance().guideList* to reach it. The benefit is that, as there is only one instance of this class, if it is modified from anywhere in the app, it will always be the same object being changed, meaning our data will always be consistent

throughout. In addition, we can create functions inside the singleton to retrieve specific data we might need, like getting the plans for a specific guide or fetching a plan or guide by its id.

Now we just need to display everything as we wish and for the moment, we just want to improve the PoC version by being able to show Guides, Plans and Steps in separate screens. For this we need to create the different Activities, ListViews and adapters to make it possible. The desired functionality is quite simple: we show a list with all the guides; when one of them is clicked we will be taken to another list with the plans that belong to that guide; when we click in a plan, we show the corresponding steps of that plan. Once implemented this is the result:

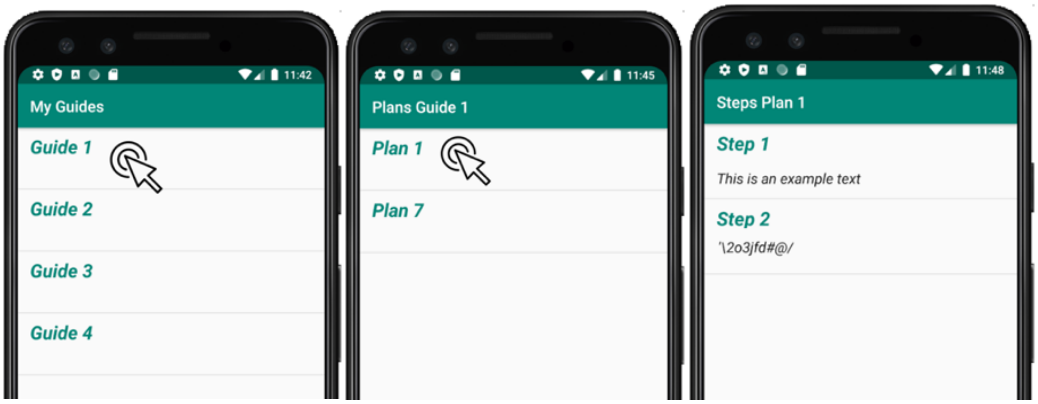


Figure 23 - Iteration 1 android functionality

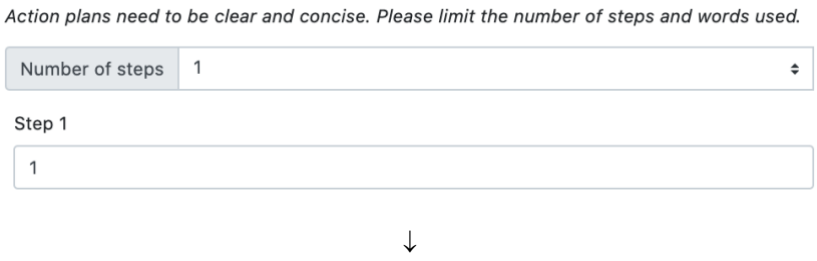
With this, we have done everything planned for this iteration including some extra features to the Creation Platform like adding a navigation bar, providing some frontend validation to our forms, alerts with the upload results, etc. and in Android, we took some steps to create a more solid foundation as the utilization of the singleton pattern and a splash screen to get the data asynchronously. This iteration took five more hours than expected, making a total of 35 hours.

5.2.2. Meeting with the client

After presenting the work done to the client, we run the acceptance tests to verify we have accomplished all our objectives. Only one test failed:

- While creating Action Plans, choose to have one step and write "1" as its text description. Then change number of steps to two. "1" should still be written in Step 1.

In this case, every time we choose a different number of steps, the inputs are created again, therefore they appear blank and the previous steps written are lost.



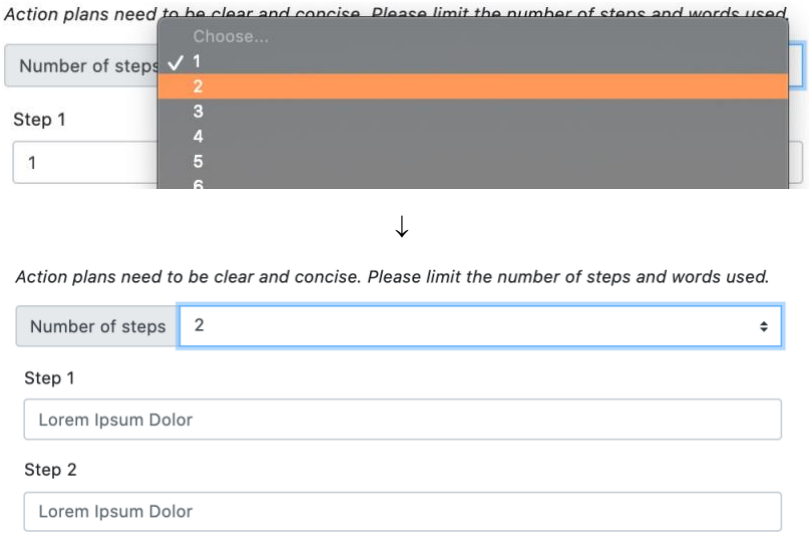


Figure 24 – Failed acceptance test - Changing number of steps

This problem is noted down and will be taken into consideration for future iterations. Having reviewed everything, the following observations were made by the client:

The Creation Platform can be improved with the following changes:

- The creation process should be more dynamic, especially with Action Plans. For example, asking for the number of steps is not a good option, since it is very probable they will not know the exact number before writing them. In addition, it should also be more flexible in options, like including the ability to change the order of the steps or delete them.
- It also seems necessary to implement a general view both for guides and plans where all content can be managed. This means creating new ones, viewing, editing, deleting them and where any new functionality as searching/filtering the content may be included.
- We can also create a landing page as a home screen, where we will have the option to change to those general views we have just mentioned.

The Android app can be improved by the following:

- Right now, all guides and plans are downloaded every time we start the application, and this can be greatly improved, especially as the available content grows in size. For this reason, we should study the different caching options available, with tools provided by Android and/or in conjunction with solutions we can develop ourselves. We should limit as much as possible the amount of times we fetch for data to be more efficient.

To encompass everything previously said, the following US are written:

- US10 – General view for guides and action plans to offer at least CRUD functionality.
- US11 – Home screen.
- US12 – Caching in Android app.

Please find these and new User Stories written in future Iterations in the corresponding Annex section, in this case, Iteration 1 - New User Stories. As the creation forms will drastically change in the next iterations, including more options as other media types and extra functionalities, we will note down that the User Experience should be improved, but we will not write a US about it until we have a clearer vision of the result we want to obtain.

5.3. Iteration 2

In this iteration we will work in the following User Stories:

- US5 – Improve Action Plan creation to allow steps with images and videos.
- US11 – Home screen

In summary, we want to improve the Creation Platform by modifying the previous Action Plan form to allow the use of images and videos in its steps and they must be seen correctly in the mobile phone. We will also create a simple home screen as landing site.

Now we will divide each of these stories into smaller tasks and write the corresponding acceptance tests. Please find all of this in the Annex section: Iteration 2 - Tasks and Acceptance Tests.

5.3.1. Development

We decided to start with the home screen creation since it should not take us much time. One of the advantages of using Bootstrap is the availability of many free templates we can use, so we searched online for one that could serve to our purpose and we found this example online. After downloading the source code, we could then adapt it to our project and this is the result:

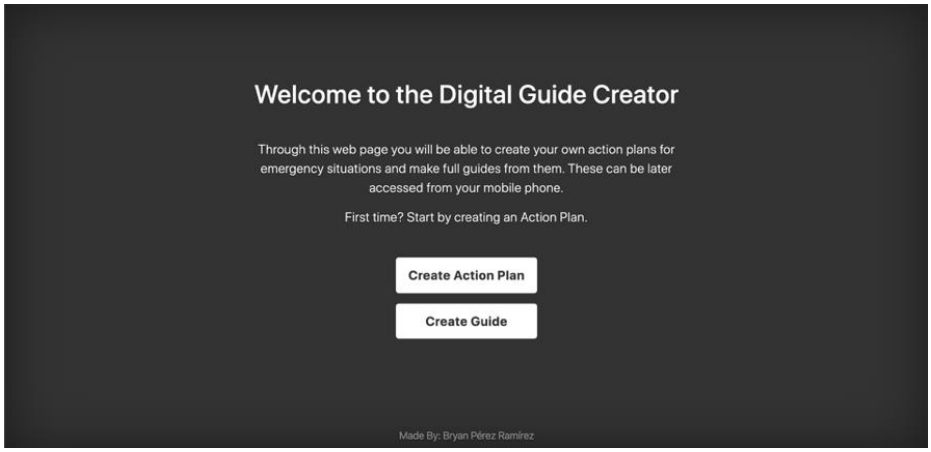


Figure 25 - Home screen

We also updated the navigation bars of other scripts so to redirect us to this page. Now we can move on to the development of the US5 and as a starting point, we decide to make the necessary changes to allow the creation of an Action Plan whose steps are composed of only images and we will work on videos later. To do so, we update our database's tables in order to hold the new media types:

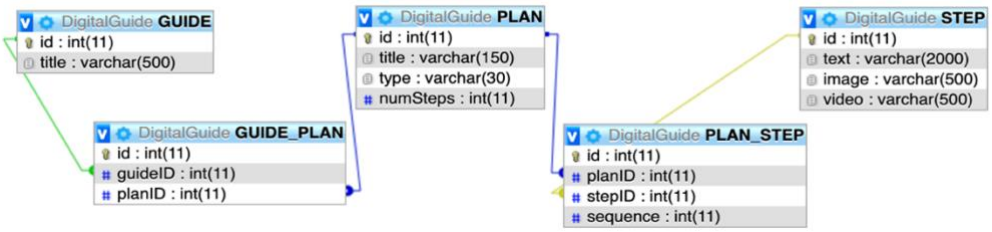


Figure 26 - Database schema - Iteration 2

First, we create two new columns for the STEP table: *image* and *video*. To make the insertion and manipulation of steps easier we decided that the columns *text, image* and *video* will be NULL by default. To PLAN we added: *type* and *numSteps.* The *type* column will be used to differentiate plans made of only text steps from the only images ones and we included *numSteps* since we believe it can be useful to have this count in Android.

Now to the create Action Plan form we add a new section so that the user can select the type of steps they want and in relation to their choice we will ask for one input or the other after they select the number of steps.

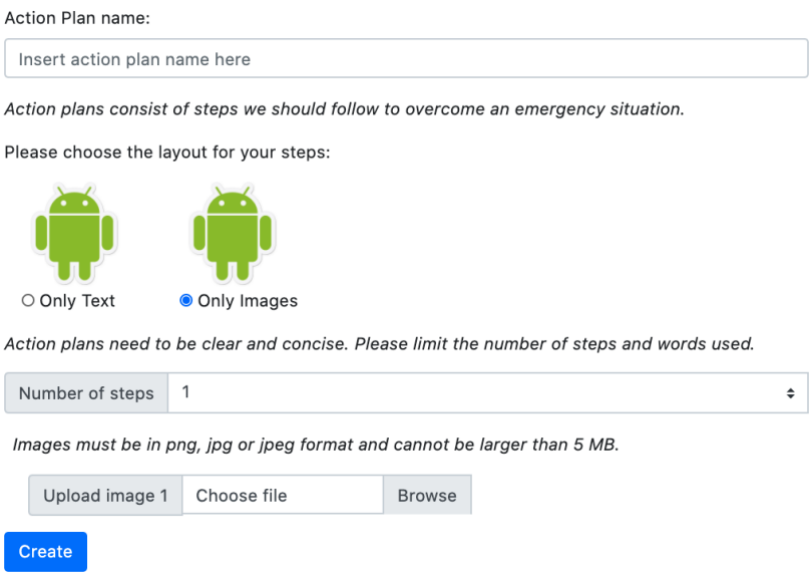


Figure 27 - Action Plan Creation - Iteration 2

Now we must update our POST script to adapt to these changes but as we had never uploaded images before we had to investigate how it could be done and the best way to do it. The two main options we found were: directly save the media in the database in BLOB format or store them in our filesystem and save the path to them in the database. Both options have their advantages and disadvantages but the general opinion we found in forums [30] [31] is that the BLOB option generally has worst results since it is specially taxing for the DB. Furthermore, if the application manages large amounts of images, the file path option turns to be more optimal, and if we also take into consideration that we will not only save images but also videos, this looks like

the best option for us too. In order to support this, we create a new folder called *uploaded* in our local project directory, where this media will be stored.

Normally, when we create a form whose method is equal to "post", the values of the different form inputs can be retrieved after submission with the PHP variable *$_POST* and as it is an associative array, we can find the aforementioned values by for example writing *$_POST["planTitle"]*. We assumed it would work in the same way with images, but it did not: uploaded files are captured in the variable *$_FILES* which includes by default some attributes that will become very handy to manage them. If for example the name of our image input is "userfile" we can know its original name on the client machine by doing *$_FILES['userfile']['name'],* its MIME type with *$_FILES['userfile']['type'],* and its size with *$_FILES['userfile']['size'].* With this information we found an example script online of how to upload images [32] from which we extracted and adapted some lines of code to create a function to determine if the uploaded images where "valid". To us this meant that the image was of one of the following MIME types: jpeg, jpg, or png and that they did not exceed a maximum size of 5 MB.

```
function checkValidImage(string $fileName){
    $max_size = 1024 * 1024 * 5; // 5 MB
    $validextensions = array("jpeg", "jpg", "png");

    $temporary = explode(".", $_FILES[$fileName]["name"]);
    $file_extension = end($temporary);

    if ((($_FILES[$fileName]["type"] == "image/jpeg") ||
         ($_FILES[$fileName]["type"] == "image/jpg") ||
         ($_FILES[$fileName]["type"] == "image/png")) &&
          in_array($file_extension, $validextensions)) {

        if ($_FILES[$fileName]["size"] < ($max_size)){
            return true;
        }
        else {return false;}
    }
    else {return false;}
}
```

Figure 28 - Check valid image function (Adapted from [28])

To move the image the user posted to our *uploaded* directory, we can use the PHP function: *move_uploaded_file(string $filename, string $destination)*. When developing this feature, we noticed that if two people uploaded a file with the same name, we would have a problem, since the last one would overwrite the previous one. To avoid this situation, our first thought was to change the image name to the unique ID we get from the database after inserting the step, but we realized that we were losing the extension of the file so, for simplicity, we decided to prepend the ID followed by a hyphen and then the original name of the file. A resulting image name after this process could be: *35-myImage.png.* Making this change will also help when fetching these images since we can provide the id of the step and look for the file that starts with that number.

We code all of this and finally upload our first plan composed of only image steps. The next challenge is to show them in the content Visualizer. Fortunately, this is a problem many people had to face before us and as a consequence there are already existing solutions that will make

this process more straightforward. Nevertheless, we might as well study the different options available in order to decide which is the best option for us. The two more used image loading libraries we found were *Glide* and *Picasso* which produce the same result but have different characteristics that set them apart and which we will study in the following table:

	Glide	**Picasso**
Dependencies	3 (2 for Kotlin)	1
Size	659 KB	106 KB
Number of methods	2678	849
Easy to use	Yes	Yes
Caching	Adaptive	Only max res version
Loading from cache	Faster	Slower
Extra features	GIFs, customization, animator, etc.	Better image quality

Table 6 - Glide vs Picasso

Although *Glide* has many extra features we might not even use, the fact that it is able to show GIFs is a characteristic of enough weight to make it our preferred choice. In addition, its caching methods are much more efficient than *Picasso's* since it stores the already resized view of the image instead of the full resolution one, which also means it loads them faster since *Picasso* resizes them each time they are loaded. For this reason, and given that both are really simple to use, we will go with our preferred option *Glide.*

To load images with *Glide* we have to provide a URL that returns it, this means we have to develop our own API which goes into our file system, searches for the folder *uploaded* and inside it an image that starts with the id we provide. We will call it *getImageById.php.* After some failed attempts and investigation, we found a solution on *StackOverflow* [33] that did exactly what we wanted, and as it could be done a bit simpler, we even proposed an edit to the original answer.

Now back in Android Studio, we update our data models to apply the changes done to the database and as we have a new attribute to know if the plan is composed of only text or images, we can create the logic to differentiate one from the other. To make it somewhat dynamic, we updated the Layout used for each row of the Steps ListView by having one TextView and one ImageView. When we load a given plan, if the *text* attribute is equal to NULL, we set the visibility of the TextView to *GONE* and the ImageView to *VISIBLE.* If instead, *image* is set to NULL, we do the opposite. The difference between *GONE* and *INVISIBLE* is that both will hide the View but only *GONE* will also hide the space it would have occupied. If we then use the Glide functions to load our desired image, this is the result we obtain (right):

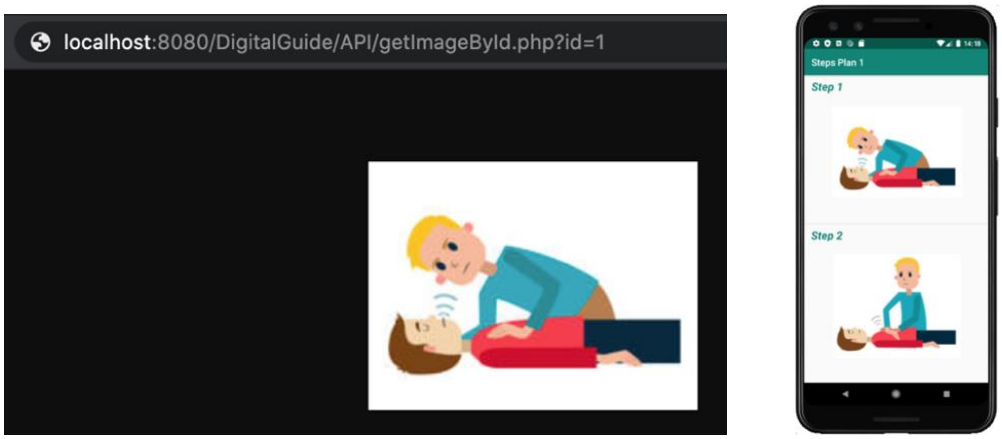


Figure 29 - API getImageById response (left) and Action Plan with images - Iteration 2 (right)

With this done we have to follow the same process but with videos. The good side is that most of the logic needed is very similar to what we have already developed for images. First, we create new inputs in the Action Plan form for the only videos option and to verify their validity when uploaded, we verify its MIME type is mp4 and the max size is 15 MB.

In relation to tools that could help us in playing videos in Android, on one side, we have the default option of using MediaPlayer and on the other, there are third-party alternatives like Vimeo, *LibVLC,* or *Vitamio,* but none of them compare to *ExoPlayer*, a media player library developed directly by Google. If you have used applications such as *YouTube* or any Google's video streaming apps, you have already experienced *ExoPlayer*. Its main features are the support of Dynamic Adaptive Streaming Over HTTP (DASH), smooth streaming, encryption of the played video and probably the most interesting one, the customization of the media control features. All of these aspects are great, but it has one main disadvantage: its implementation is a little more intricate. Its extra features are nice bonuses, but as for the moment, we just want to test how videos work and not spend too much time in features we might not even need, we will work with the default MediaPlayer for now, but will certainly consider implementing *ExoPlayer* in the future.

As it was the case for images, we also need to code an API that searches for the video in our filesystem and expose it through a URL. In this case the result is also similar to images but with the inclusion of some extra headers for it to work correctly. In Android Studio we just need to do the same process but now using VideoView instead of ImageView. After setting it up, we also attached a MediaController component to the video which will allow us to play/stop, rewind and fast forward the video, as well as the inclusion of an interactive progress bar.

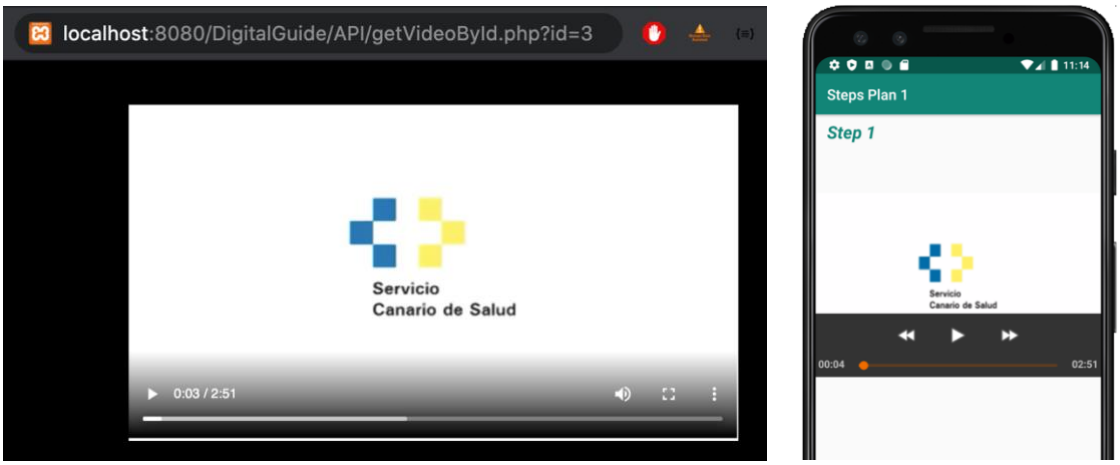


Figure 30 - API getVideoById response (left) and Action Plan with videos - Iteration 2 (right)

Due to the novelty of things done throughout this iteration some extra hours of investigation were needed and coupled with some complications we also had to confront, we underestimated the time we would take to develop US5. On the other side, we also developed US11 faster than expected, which meant that the total amount needed was of 35 hours, five more than calculated.

5.3.2. Meeting with the client

After presenting the work done to the client, we run the acceptance tests to verify we have accomplished all our objectives. All of the tests passed except one:

- Create an Action Plan with two images, one valid and the other one invalid. Action Plan and images should not be saved. Not valid alert should pop up.

In this case, the Action Plan is not uploaded, and the correct alert is showed, so this works as it is supposed to. Nevertheless, the valid image was uploaded to our filesystem. This is because instead of checking that all the images were valid before uploading, we just checked them individually, so the valid ones were submitted.

Having reviewed everything, the following observations were made by the client:

- Apart from the current ListView we should have at least another way to show Action Plans in Android. The option the client would like to see, is a screen where we show one step at a time, and we can change between steps with back and forward buttons. In this way we can put more emphasis in each of the single steps.
- In relation to videos, we should think about how content creators will upload them. In this case, adding the option of providing a URL to a *YouTube* video rather than requiring them to upload the video itself, looks like a better alternative. We should study this option and the best tools to implement it, probably being ExoPlayer the best solution.

To encompass everything previously said, the following US are written:

- US5.1 – Fix Action Plan creation so that all images/videos have to be valid to upload.
- US13 – Buttons view for steps.

- US14 – Improve options to upload videos in Action Plan creation.

Please find their full description in the Annex section Iteration 2 - New User Stories.

5.4. Iteration 3

In this iteration we will work in the following User Stories:

- US5.1 – Fix Action Plan creation so that all images/videos have to be valid to upload.
- US7 – Improve Action Plan creation to allow other media types, customize their layout in Android and add extra options.
- US13 – Buttons view for steps.

In summary, what we are going to do in this iteration is: first, fix an error from the previous US5 so that it works as it should. We will also tackle US7, which is one of the most important User Stories of this project. The objective is to find how we can allow the customization of steps so that creators can generate Action Plans exactly how they want them to. This is a challenge both in the Creation Platform as in the content Visualizer since on one side, we have to create the tools to make this customization possible and in the other, we have to adapt to every single possible result and visualize it properly. We will also develop US13, where we will create a new way to navigate the steps of a plan.

Now we will divide each of these stories into smaller tasks and write the corresponding acceptance tests. Please find all of this in the Annex section: Iteration 3 - Tasks and Acceptance Tests.

5.4.1. Development

We decided to start with US5.1 since it is a pretty easy fix: we just create a for loop where all uploaded images are checked. From the moment one is not valid, we break the loop, finish the process and just pop an alert to notify the user. If all are valid, we can proceed with the upload.

Now we are going to develop US13, since it does not look very complicated either. What we want to create is a screen that shows two buttons, one at the left and the other one on the right, that when clicked will show a different step. As what we are trying to develop in this story is just the logic of changing steps with the buttons, we are not going to support the option of only videos at the moment, we will try with only text steps first and then to make sure it works, we will also test it with images. To do this we will follow the same approach as before, for only text steps we hide the ImageView and for the opposite case, we hide the TextView.

First, we create a new Activity and a simple layout with two TextViews for the title and step text, and two *Floating Action Buttons,* which in Android are just regular buttons but that will always be on top of the content. Then we just need to create the logic to change steps, but here we noticed that these buttons should not be clickable all the time since if the user presses the next button and there is no next step, we will get a null pointer exception and the app will crash, one of the worst things that can happen in an application. The same will happen with the other button of

course, so the logic should be: if there is only one step, no buttons should be displayed; if there is no previous step from the current, back button should not be displayed and the same with the opposite case; in any other cases both buttons should be visible. Furthermore, every time we click on one of the buttons, we need to update the step showed in the screen, as we have all the information in our singleton, this will not be a problem. After developing it, this is the result:

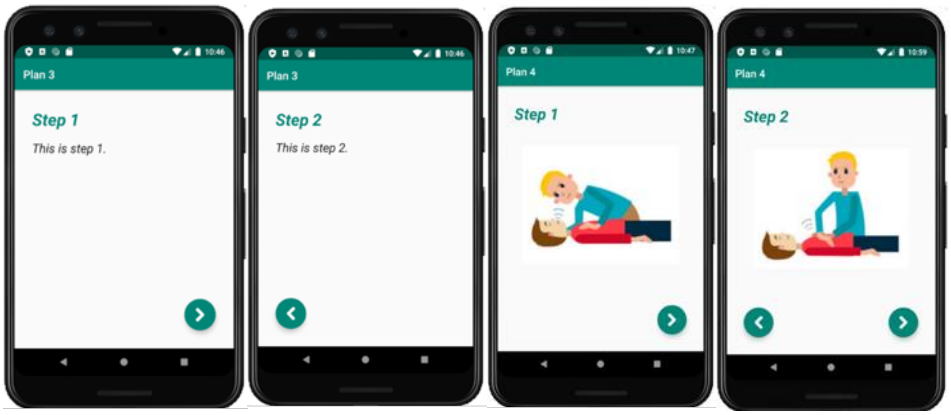


Figure 31 - Buttons View - Iteration 3

To dive into US7, we might start by looking into how to include GIFs in our plans and as this is one of *Glide's* basic functionalities, it should be easy to implement. After some searching, in previous Glide versions (>3.0) we needed to add the line *.asGif()* but as we are using version 4.9 Glide detects the extension itself, so it works exactly the same as with images, no extra code is required. For this reason, in the Creation Platform we will treat GIFs exactly the same way as images, we just need to update the check validity function to also accept the *.gif* extension. We made a quick test and it works like a charm, where we did not even need to modify the API to retrieve images as it was prepared to return different image formats automatically.

We proceed with the most important task: allowing layout customization. As we are not completely certain about how to do this, we may start by designing the result we would like to achieve. From our State-of-the-Art study of the apps which are already in the market, we found out that Action Plans consisted of simply text, images and in some cases videos, but how they were arranged in the layout varied from one case to another. With this in mind we designed some of the possibilities the creators may want to achieve:

Figure 32 - Steps possibilities design

From this simple design we learnt that the amount of possibilities to arrange the different inputs in groups of 1 to *n*, where *n* is the number of inputs we offer, and the order of the elements does matter, can be determined by the formula: $\sum_{r=1}^{r=n} \frac{n!}{(n-r)!}$. So, if for example, we have the options of text, images and videos, *n* will be equal to three and there would be a total of fifteen possibilities. If we added one more input type, the possibilities increase exponentially to sixty-four. With this simple idea we are starting to understand how difficult it will be for us to allow the option of choosing which type of input they want and in which order (if more than one is chosen). The problem is even more complicated when we think about adapting all these choices to Android.

As there is no absolute answer about the best way we can offer this, we will make use of a *spike* both for the Creation Platform and the Visualizer, in order to find possible solutions for this problem. So, let's start with the web platform and make some sketches.

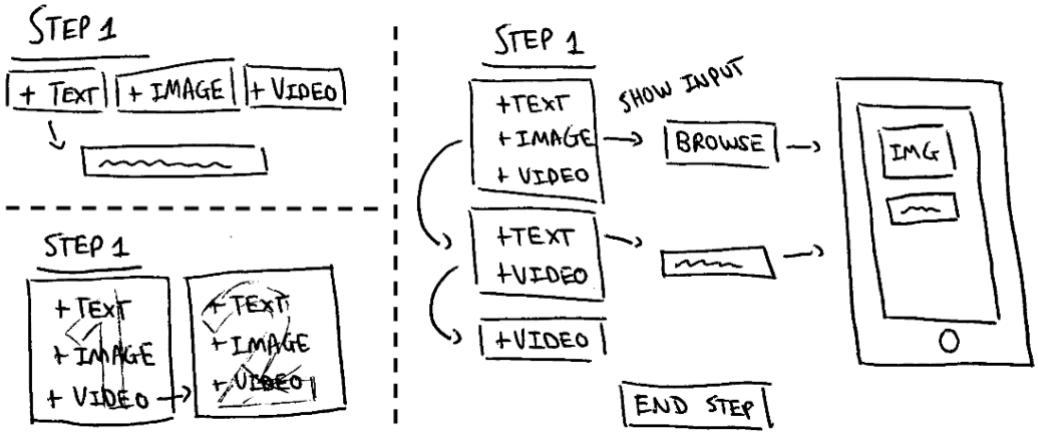


Figure 33 - Dynamic steps first sketch

The main ideas we came up with were based on giving the creator the ability of assembling steps to their liking. Looking at Figure 35, we started with a basic concept (top left) of letting them choose the input type, but it was too simple, and we were not conveying that they could be arranged in a specific order. Beneath this design (bottom left) we improved the previous concept by initially showing one block with the three different inputs. Once one is clicked, a second block would pop up, where the previously chosen input is not an option anymore. Merging these two ideas (right), we thought that a plausible option would be to have three blocks, one on top of the other, each one with the three different inputs. Once an input is chosen, the other blocks are updated by deleting that input and at the same time, a form appears to their right so that users can insert the required information. On top of this, if we managed to mock the resulting choices on a simulated phone, it would be obvious for the user that they are also organizing the different elements in the final view. We decided to implement this last concept and test if it really works.

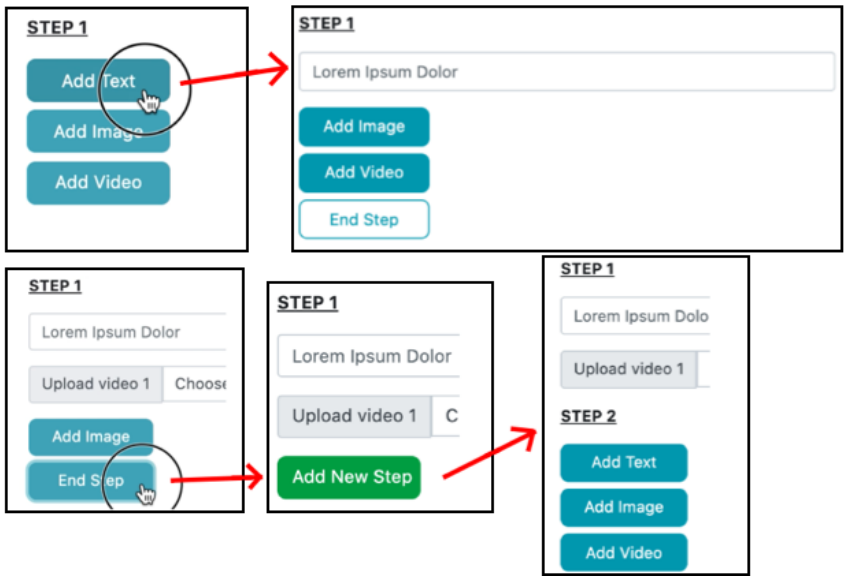


Figure 34 - Dynamic steps implementation

This is the result we came up with, which looks similar but has some slight variations. On top we can see that instead of showing three blocks, we just show one, where the selected option is converted into the input and the other options remain unchanged. At the same time, we show a new button at the bottom to end the current step. If in turn, this button is also clicked, we provide the choice of adding a new step, which basically starts the process all over again.

This approach may work but it has its caveats: first, we are not really sure the user will understand they are choosing the arrangement of the elements, second, with the current skills we have in frontend programming and the time budget, developing the simulated view of the result in Android will certainly be too complicated and time consuming, third, we would also have to find a way to finalize the process, always making sure that the creators do not upload invalid info and fourth, giving the creators options to rectify like undoing an input choice or changing the order they first established will be very problematic. As we are blocked with how we can improve this, we are going to move onto the other spike and we will come back to this later.

In Android Studio, one of the first options we thought about was using *fragments*, which in simple words, are small components with their own logic and UI that can be used inside Activities, which basically makes them sub-activities. An activity may have multiple fragments at a time, usually defining a portion of the UI but with the advantage they can be reused in other activities. With this in mind, we could have one fragment per input and show or hide it depending on its value. But in order to have each input in the three different positions: top, middle and bottom, we would basically need to have three fragments in each of the positions and show one and hide the rest accordingly. This does not look very efficient, so fragments are discarded for the moment.

As we have already mentioned, an Activity is associated to a Layout which basically is the UI for that screen. An option we also considered was creating a code able to generate the Layout at

runtime (instead of using an XML created at design time). This was actually achievable, but the problem is that the required code would end up being very dense and inefficient, because it is not just creating each of the different elements, it is also necessary to: uniquely identify them, give them a position in the screen, give them a style, set their contents, etc.

As this was not a good alternative either, we came up with two other different possibilities. One option is to do what we proposed with fragments but just do it in the layout where we would have the three different positions and in each of them have a TextView, ImageView and VideoView and just display them or not depending if their value is NULL. This would work but is not a very optimal solution. The other option is to create a layout per each of the possibilities and in the ListView adapter, inflate one or the other, again depending on the values. This option although more organized is not very optimal either, since the interesting part of ListViews is that we could recycle the already created rows and reuse them, but if we change the layout per row, we would have to generate them again each time, omitting their whole purpose. in addition, as we calculated before, if we just offered three different input types, we would need to create fifteen different layouts, but if we would like to grow in input options this would be unmanageable.

With these spikes we learnt that offering this flexibility will not be an easy task, so we decided to do a half-iteration meeting with the client in order to decide what approach we should take. In relation to the Creation Platform spike, the client also thought that we were not adequately conveying that we were also choosing the order of the input types. For it to possibly work we would need the simulation of the result in the phone next to it, as per design, but he also noted that this would practically be the same as creating a GUI that let you build the steps directly in a phone-like simulation for example, and that was not really the objective of the project.

After some deliberation we concluded that offering this complete flexibility did not make sense since it could be an entire project by itself, so he suggested that instead of allowing creators to assemble their own steps, we could offer them a selection of different possibilities and let them choose from that closed set. In this way, in the content Visualizer we would just need to create the different layouts that correspond to the options we offer and use them accordingly. As we were already on the topic, we chose the initial set of combinations we would like to offer based on what we feel will be the more common choices:

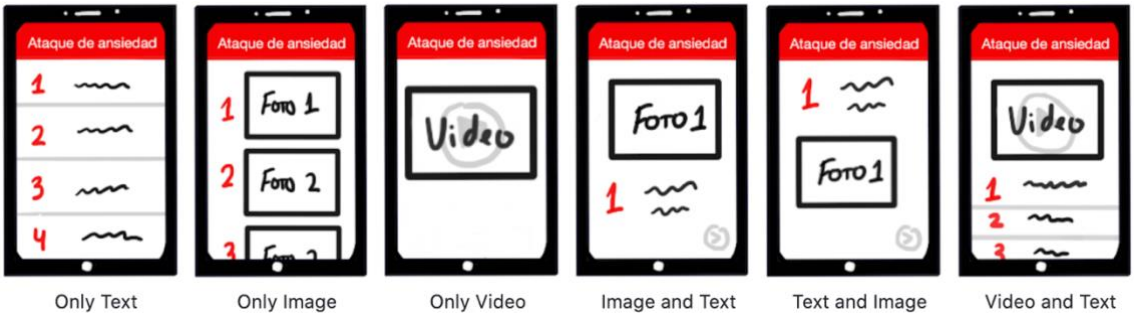


Figure 35 - Set of layouts

The idea is to add a section to the Action Plan creation form where these images will be displayed as possible options to select (we will use these images as reference until we design the final ones), and when one is clicked, show the input that corresponds to the option selected. We can also leave the image to one side of the input so that the user can relate that the information they are inserting will be displayed on that position in the Visualizer later. Before this, we could also include another section to choose how to navigate the steps. For the moment, we will have two options: *Scroll* up and down through the steps or use the *Buttons* to change between them. In the future, this section can include new options such as an expandable list, where each step can be closed or opened to show it contents, or another one where buttons can be replaced with other options like tabs or swiping from one to the other.

With all of these options, the client also showed his concern about having a web page that grows too much vertically and that right now we have too much explanatory text. With this new perspective, we added new tasks to the initial ones of US7:

- Update Action Plan creation form to choose steps navigation: Scroll or Buttons.
- Update Action Plan creation form to choose from the step types from *Figure 37.*
- When one of these are chosen, show the corresponding input of the type: If its only text, include a TextArea input, if only image, an image input type, etc.
- Investigate ways we can make the Action Plan Creation form to occupy less vertical space.
- Reduce the explanation texts of the form. Investigate other ways we can convey this information without being too interfering.

To avoid having too much explanatory text, we can use HTML *tooltips,* which Bootstrap also has its version of. Tooltips are basically small pop up boxes that are showed when the user hovers over the element that contains it, in our case, a button with the question mark symbol. For the other problem we first tried to include a button next to each section that basically collapsed the input but then, we found an option that looked nicer and could solve the problem and it is Bootstrap's *Accordion* [34]. Its use is very simple: there are these components called *cards,* that when clicked, they open or hide its contents. If another card is clicked, this one opens and the previous one closes, creating this accordion-like effect; nevertheless, this functionality is optional.

In the case of the Action Plan form, the first card will be for its title, the second for the navigation and the third and following ones for the steps. Inside these step cards, first, we will have the selection of step types we defined on *Figure 37* and then, when one is clicked, the corresponding input will be shown. Finally, we must include the option to add new steps. We deliberately chose the set of step types we will offer, a total of six, but these may grow in the future. As they have to be showed inside the card, there is a limit of elements that can be seen at a time, and in addition, they can be further limited by the user's screen resolution.

With this in mind we developed a simple solution: the different step types are now radio buttons and we use the CSS *overflow* property set to *scroll* so that if the content inside the card is larger than itself, a slider will be displayed at the bottom to move through the content freely.

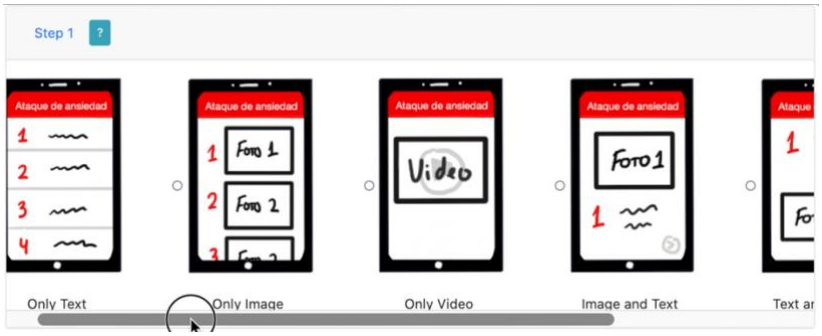


Figure 36 - Step type selection slider

We then proceed to create all the views for when each of these options is selected. Once in this new view, we also included a back button so that creators can go back to selection if they wanted to. At the bottom right of the card we also added the buttons for adding new steps and to remove the current one. At the moment, the delete button is only available from step 2 onwards, since we do not want users to delete all steps and are stuck with no way to include them back again.

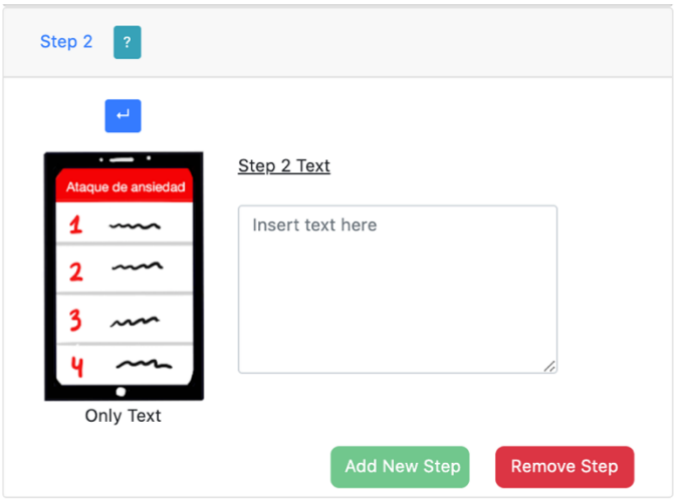


Figure 37 - Step creation - Iteration 3

To include the adding new step functionality we needed to make this process very dynamic. When the button is clicked a new card with the step type selection inside it must appear beneath it. Then, each of these types has their own required input and as they can be chosen for more than one step, their ids could not be fixed, so we made them to always depend on the step number. For modularity, each of these aforementioned functionalities are coded in their own separate method, where it will be really easy to add new options in the future.

As we have already surpassed the estimated hours for this iteration, we will not have time to update the backend to upload this new type of Action Plan but at least, we want to verify that we are able to capture with the $_POST variable all of the inputs we have created. After fixing a

problem where the type selection was being overwritten with the corresponding inputs (which meant we were losing this value) we could close this iteration. As we can see, there was much more work than initially expected, needing to spend 10 more hours, making a total of 40.

5.4.2. Meeting with the client

After presenting the work done to the client, we run the acceptance tests to verify we have accomplished all our objectives. All of the tests passed except the ones about tasks we did not have time to develop, so we will cover these in new User Stories:

- Try different customization options and verify they are uploaded correctly.
- These should be correctly represented in Android.
- Create an Action Plan with the option of calling a number, verify that if clicked the call works correctly.

Having reviewed everything, the following observations were made by the client:

- We can make visible the different cards as you progress in the form. For example, at first, we can just show the card that asks for an Action Plan name. We can add an OK button to it that, when clicked, will proceed to close the current card, open the next one and make it visible. In this way the process is more dynamic and more user friendly.
- For the step navigation we can show GIFs of examples with the Scrolling and Buttons functionalities so that its clearer what they are choosing.
- The position of the buttons for adding and removing the step are not adequate. His suggestion is to have three buttons at the bottom outside the cards to: Save and Cancel the Action Plan and separate from these, the one to Add the new step. The remove button can remain on the card but probably somewhere else so that it is not clicked by accident. Furthermore, we should be able to delete other steps not just the last one.
- As we have implemented the accordion and its cards close as you fill the form, it will be nice to see in some way what information you have inputted so that creators can know what they inserted without the need to open the card again.
- We have put a lot of emphasis on Action Plans being composed of Steps but what if a plan is just a series of considerations or instructions without specific order? Maybe it's an Action Plan that shows where the extinguishers of the USJ are at, for example. For this reason, although most of the times it will be composed of steps, we should allow to give each "step" a title and in that way, users can have as title different values than Step 1, 2, 3… Us as developers will still call them steps to avoid confusion but note that they do not really have to be steps per se.
- Fix navigation bar to the top so that although we scroll down, we can always see it.

To encompass everything previously said, the following US are written:

- US7.1 – Fix Action Plan creation to improve UX, update backend and Visualizer.

- US7.2 – Create new step type to include phone calls.

Please find their full description in the Annex section Iteration 3 - New User Stories.

5.5. Iteration 4

In this iteration we will work in the following User Stories:

- US7.1 – Fix Action Plan creation to improve UX, update backend and Visualizer.
- US8 – Login / Authentication system

In summary, we want to modify the Action Plan creation form with a series of improvements in order to revise the UX. Then we have to update the POST script to upload these new types of plans and when this is ready, we will have to also update the content Visualizer to adapt to all of these changes. When done, we will develop a complete register/login system.

Now we will divide each of these stories into smaller tasks and write the corresponding acceptance tests. Please find all of this in the Annex section: Iteration 4 - Tasks and Acceptance Tests.

5.5.1. Development

We decided to start with the development of all of the tasks related to improving the UX but as many of them are straightforward, we will not make comments about all, but only those that are interesting or require some investigation. We will then take a look at the resulting product.

The first comment is related to the Cancel plan button and its functionality: we felt like it needed double confirmation since creators could lose all progress done if they clicked it accidentally. To solve this, we first show a modal to make sure the creator really wants to cancel it.

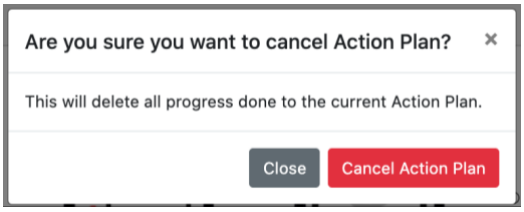


Figure 38 - Cancel Action Plan modal

Cards in the accordion close as others open so if you want to know what you wrote inside, you would have to open them again. To avoid this, we implemented a simple solution: the title of each card will be updated with the key information inserted in each case. For step cards we display their title, since even though they contain more information it is enough for creators to keep track of them. We can see this result in Figure 39.

Talking about step titles, we decided to set it by default to "Step" plus the number of the current step since it is what creators will probably use the most. Nevertheless, it can be modified, and any title can be inserted, given them full customizability of the Action Plans. As we also allow the deletion of steps, if titles are not changed, we could have steps whose titles are Step 1, 5, and 8, but as this can be corrected by the user, we leave this on their hands to fix it. As in the future

we will provide means to edit plans, if they do not notice at first, they can always go back and solve it later. After applying all requested changes, this is the result we obtained:

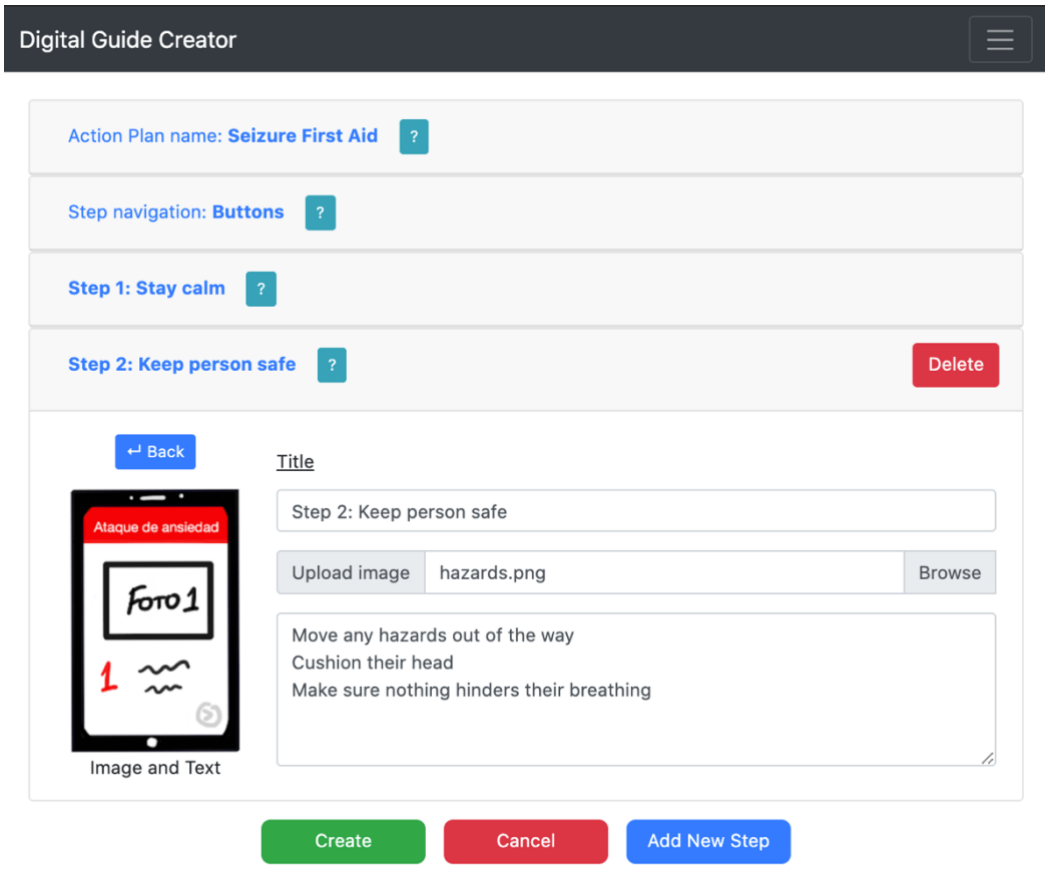


Figure 39 - Action Plan creation form - Iteration 4

Now we have to renovate the backend and the first thing to do is update the database.

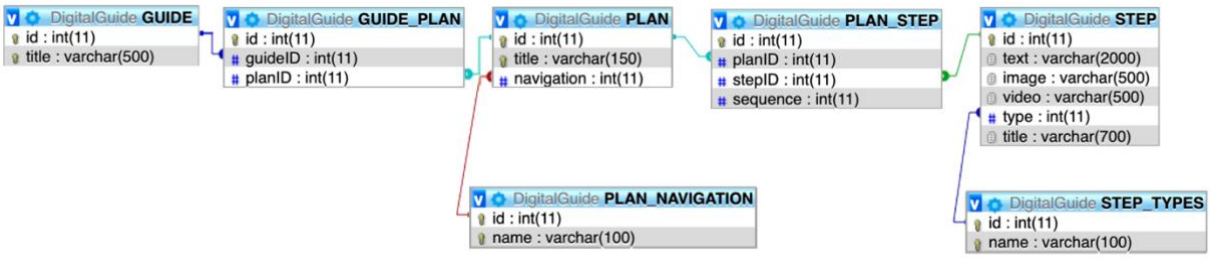


Figure 40 - Database tables - Iteration 4

The changes done were:

- Removed *numSteps* from PLAN since we verified we did not need it in Android.
- Removed *type* from PLAN and introduced *navigation* which is related to a new table called PLAN_NAVIGATION where we will store the different types of navigation we offer. For now, the only two options are Scroll and Buttons, so we inserted them manually.

- We added to STEP two new columns: *type* and *title. Type* makes reference to a new table called STEP_TYPES which holds the different options of step types we offer, which are six for the moment but may be expanded in the future. We insert them manually.

With these changes we still maintain the third normal form standard of relational databases.

	id	name
Editar Copiar Borrar	1	Scroll
Editar Copiar Borrar	2	Buttons

	id	name
Editar Copiar Borrar	1	Only Text
Editar Copiar Borrar	2	Only Image
Editar Copiar Borrar	3	Only Video
Editar Copiar Borrar	4	Image and Text
Editar Copiar Borrar	5	Text and Image
Editar Copiar Borrar	6	Video and text

Figure 41 - PLAN_NAVIGATION (left) and STEP_TYPES (right) rows

We are ready to upload these Action Plans, but first, we have to take something into consideration: previously, we only allowed the deletion of the last step. This meant that we knew that our cards would always be in numerical order, starting from one and ending in *n, n* being the total number of added steps. In order to differentiate the input of one step from another, we always added the ID of the card to the input (e.g. step1Text, image1, etc.). Therefore, finding them in the $_POST variable was simple since just by knowing the total number of steps we could then find all the input we were looking for. On the other hand, now we allow the deletion of any step, which means that the resulting cards could be any combination of IDs (e.g. card 1, 5 and 7) so we require some extra preparation before we can start uploading everything.

To find how many steps there are and their associated ID, we will have to find inside the $_POST variable how many of the submitted inputs start with the name "radioSteps", variable that holds the step type selection for each step. This variable has appended the ID of the card at the end like "radioSteps1", so what we will do is each time we find it, get that ID and add it to an array. Now we just need to iterate this array and with each ID we will be able to find all the values.

Therefore, the final uploading algorithm will be the following (input is processed first for security):

1. Find how many steps we have, and their IDs as explained above.
2. Upload the PLAN and get its ID.
3. Iterate each STEP to:
    a. Get its type.
    b. If it contains media, check it is valid.
        i. If not, exit the loop and delete PLAN.
    c. Upload the STEP and get its ID.
    d. Insert new row to PLAN_STEP.
    e. If it contained media, store it in our filesystem with prepended STEP ID.
        i. Update STEP to store the new name.

We developed the code to make this work and when done, tested it to make sure it was finished. At this moment, the opportunity of testing this platform with a real user arose, and it was the perfect moment to do so since we just had finish the renovations.

For this meeting we will use two main tools: *Skype* to make the call and *TeamViewer* so that they can access our computer and test the website. When everything is set up, we will make a short presentation about what the platform is and what it allows you to do. Then, we will let them use the platform freely, telling them to please think out loud in the process. Meanwhile we will be taking notes about comments or suggestions and of the behavior using the platform. With this prepared, we established a meeting with Ana Ramírez, member of the GTC group and teacher of the institute *IES Lomo de la Herradura* in Gran Canaria. After the session, we compiled all the feedback received and summarized it into these key points:

- User directly clicks on card title (styled as a link) although card is already opened.
- About the navigation GIFs, user states that once you stop and look at the differences between them, it is clear what each option is about, but maybe the action of scrolling or buttons could be made more apparent by highlighting it with more eye-catching colors.
- In steps, combining the resulting example image in Android with the corresponding input is a good idea, but these should be as similar as possible, because user is very much guided by these images.
- There is a big problem with the image input and it is that we used Bootstrap's default example which included three sections:

Figure 42 - Bootstrap default file input

  If user clicks on "Choose image file" or "Browse" the file explorer is opened but the section of "Upload Image" does not have this functionality, and user always clicks on this section.
- User is accustomed to drag and drop input for images. She would like to have that option.
- With videos, user expected to provide a *YouTube* link instead of uploading it.
- User would also the option of saving a draft, to resume the creation at another time.
- Although this is just a prototype, the user already envisioned the possibilities this could have. As ideas she proposed the creation of various plans such as a personalized plan for students with specific chronical diseases, a plan that shows the steps to evacuate the institute in case of emergency and even some plans not related to health at all, like a plan to teach her students how to follow online classes.

This was the first test of the platform and we are satisfied with the result since it did not take much for her to learn and realize the possibilities it brought. Most of the feedback we received is personal opinion, which is important for the future, but for now we cannot make changes based on the opinion of one user. Nonetheless, thanks to it we found a critical point in our system and

it was the image input, which left the user confused in many occasions, since it did not respond as expected. The good side is that it is an easy fix which we will apply instantly: we will just remove the first section of the input and leave the other two as they are right now.

Now we will start with the last part of this User Story and it is updating the Visualizer, so we will head back to Android Studio. The first thing to do as always is to update our data models to get the new attributes. As we already know, we have two types of navigation: Scroll and Buttons, and as this project is a prototype, we will take a different approach for each of them to test which is the better solution. For Scroll we will develop a solution purely based on the step types we have right now, which is easier but less scalable. For Buttons we will try to leave it more open, complicating it a bit, but creating a much more scalable result.

In the first case what we do is create a layout that has all the input types ordered in a way which we can just show or hide the others to create the step types we offer, resulting in layout like this:

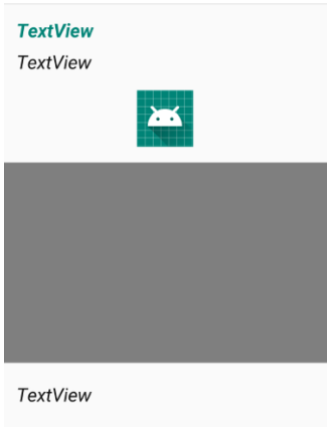


Figure 43 - Custom step layout for Scroll navigation

In it we can see in order from top to bottom: the title, text on top, image, video and text on bottom. Now in our adapter, we just have to look at the step type and provide the corresponding values to show them in a list and hide the rest, and as we can make them invisible without taking any screen space, it works as expected. In the next example we created a plan composed of three steps, each of a different type: only video, image and text and only image. The obtained result works and looks fine.

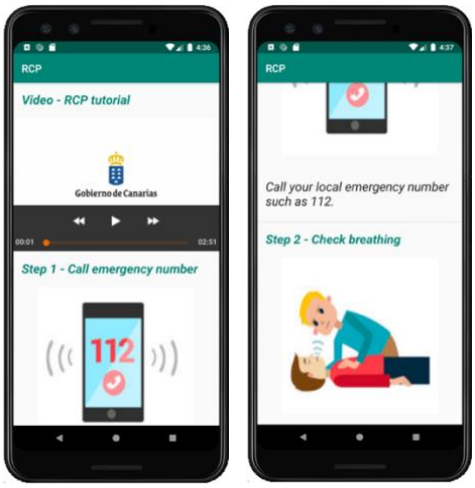


Figure 44 - Scrolling steps - Iteration 4

This works perfectly right now as it is specifically built for the current step types, but if we wanted to allow more of them, both the layout and the necessary code to make it work will get too messy very fast, so we would need to take a similar approach to the one we will do with Buttons.

In this case, what we will do is: have a base layout with the elements common to all the types, which are just the title and back and forward buttons. Then we will create another layout for each of the step types we offer and include them in that base layout but, with the visibility set to *GONE*. Then in the Buttons Activity, when a button is pressed, we will get the current step type, find the corresponding layout, set it to visible and finally, show the required information. In next button presses we also have to make sure we first hide the previous layout and then display the new one, as long as they are different of course.

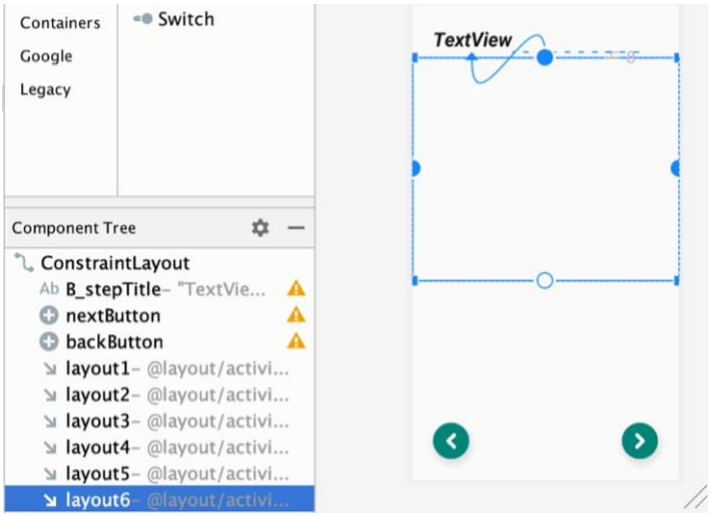


Figure 45 - Buttons base layout

This approach, although a bit more tedious, makes more sense than the other one, since it is fully scalable and adding new step options in the future will be very straightforward. Here we have an example of the same plan used in Figure 48 but with Buttons navigation:

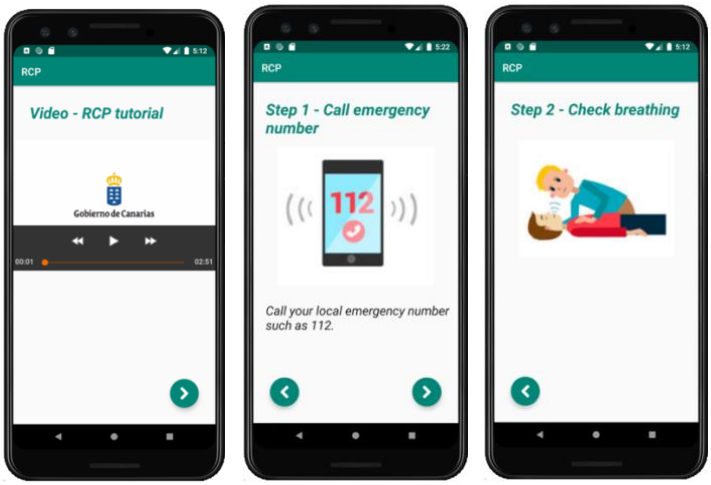


Figure 46 - Buttons steps - Iteration 4

With this we have finalized US7.1, which means we can start with US8: a register/login system. This is a very common feature in web applications, which means there are many templates online that are free to use. After some searching, we found on *Bootsnip.com* a template done by the user *syanaputra* [35] we both liked and was easy to adapt to our requirements.

After the initial setup we are ready to start with the development and the first thing we will do is change the CSS so that the background matches with the one we used in our Home page. As we are changing it from a white to a dark background, we must also modify the colors of other elements so that they can be seen correctly. Now, we will start with the register site, and after changing the default inputs for the ones we need, we include some frontend validation so that all the fields match the required criteria. To upload this information, we need to create a new table in the database and we will call it CREATOR. Its attributes are: *id, email, password, name, alias, verified, salt* and *activationCode.*

As we can see, there are some columns we have not mentioned before but we will explain what they are for as we develop this story. We have the frontend side of the register complete, so let's work on the backend part. While doing this, we will try to take some basic security precautions and the first thing we should avoid is storing passwords as plain text. We will solve this with a basic but effective solution, which, needless to say, could be more complex but it is better than nothing: we will generate a random string of ten characters with the PHP function *openssl_random_pseudo_bytes(10)* which we will call *salt.* Then, we create a variable whose value is the concatenation of the salt, the introduced password and the salt again. We will finally use a MD5 hash function to encrypt this value and the returned hash is what we store in the database. For the unaware, a hash function is one that provided a given input, usually a string, generates another unique string of a set length, but which most importantly, cannot be used to reverse the process: providing a hash value will never return you the original string.

For us to be able to verify the login information later, we have no other option that storing the salt in our database too, and this could certainly be done in a more secure way and not directly

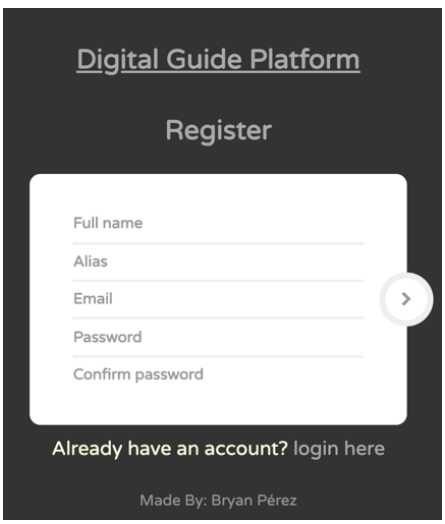
in the CREATOR table, but at least we took some basic security measures. The other values we need to upload are *verified* and *activationCode*. The first will be default to 0 as it will be used to know if the email has been verified already, when it will be set to 1. *ActivationCode* is another ten-character random string that we will use later.

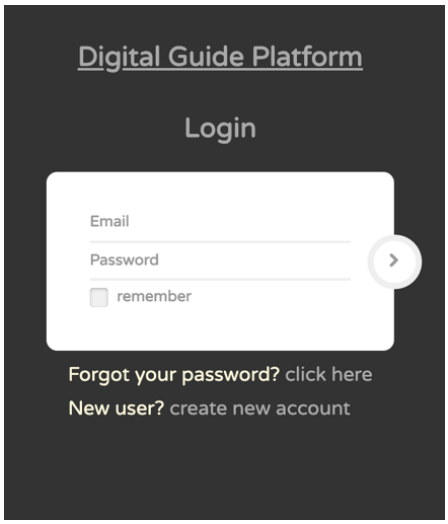
The only thing left is verifying there is no already existing user with the provided email or alias (if so, we will show an alert) and proceed to upload this creator. If successful, we want the user to verify their email account. To do this, we will send an email with a special link to the corresponding account, that when clicked will take them to another site where we will do the verifying process in the background. To make this URL unique for each user, it will be composed by the route to the mentioned script and the activationCode as a parameter. For mailing, we use the library *PHPMailer,* which is one of the standard PHP libraries to send mails nowadays.

When the user clicks on the link, we get the code from the URL, find a user with that code and verify the account if it was not already done. We also capture the cases where no user is found, or the code is invalid. If all went alright, we redirect them to the login page.

About login, the main question is: how do we know the password is correct? First, we must search for the user with the given email, then, get its salt, create the hash value with the given password and the salt (as we did in register) and finally, verify that passwords match. We also need to capture the cases were the provided data is not valid or where the email has not been verified yet, but if everything is correct, we redirect the user to the home page of the platform.

For the forget password, the user needs to provide an email to which we will send a very similar link to the one in register but pointing to another script. This one will verify that the user exists and if so, provide a new form so that they are able to insert a new password. As we are updating the CREATOR table, we use this opportunity to also generate and insert a new activation code so that the previous one cannot be used anymore.





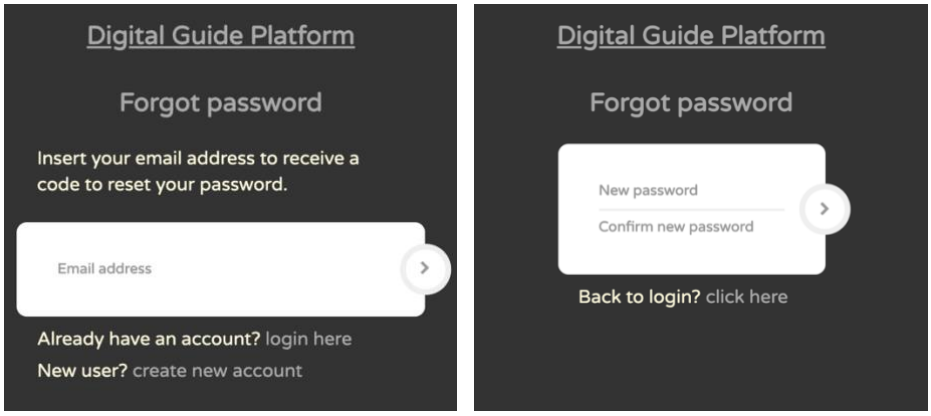


Figure 47 - Register, Login and Forget Password screens

To complete this US the only thing left to do is update the database to store who the creators of the content are and change our POST scripts to upload this information. Therefore, the tables GUIDE and PLAN are updated with a new column *creator*, a foreign key to the CREATOR table.

But now we have another problem: users must be logged in to access and use our platform but how can we ensure this and identify them throughout all the platform? We can use PHP's variable *$_SESSION*, that stores information to be used throughout multiple pages. When login, we will store the email of the user in this variable and to access it from anywhere, we create a new script that will be in charge of: start the session and check that the login variable exists, if it does not, we redirect them to the login page immediately, in this way we ensure they always are logged in. If it exists, we do a quick query to get the ID corresponding to that email and store it in a new variable, so we can always identify who the user is and operate with this value when necessary. Now we just need to include this script anywhere we want this verification to happen.

With this, we conclude this iteration, which considerably surpassed the initial estimation, especially because we were not counting the testing session and the following work related to it. On top of this, we did not evaluate correctly how time consuming some tasks were going to be, so all in all, this iteration took us 45 hours, 15 more than the expected.

5.5.2. Meeting with the client

After presenting the work done to the client, we run the acceptance tests to verify we have accomplished all our objectives. All of the tests passed except one:

- Verify that Remove Step button, removes the corresponding step and not others. With five steps try deleting the first, middle and last step. Steps left should be 2 and 4.

The way in which we developed the deleting of steps does not allow the deletion of the first step since it is fixed so that the user is never able to be left without steps. This may be improved so that all steps are removable until there is only one left. Nevertheless, the client believes it is not an important feature to work on right now, so this may be left as future work.

Having reviewed everything, the following observations were made by the client:

The client liked the renovation done to the creation form but made the following suggestions:

- Create and Cancel button should be changed to Save Action Plan and Cancel Action Plan.
- Add New Step button should be separated from Action Plan buttons to convey they are for different things: one for steps and the others for plans.
- Client would like to add icons to the buttons to anticipate what clicking them will do.

Moving forward, now that we have creators, we have to think about a permissions system where all the content created may be public or private. We will have to update our creation forms to offer this possibility and most importantly study what they imply. How do we store who has access to what content? How do we share private content? How does this affect the Visualizer? All are questions we must answer. The client also wants us to develop the "general" views for guides and plans, where we will be able to create new ones, show, update the already existing ones and delete them. Paired with the permission system we mentioned above, we also want creators to access other users' content, so permissions must be taken into consideration.

To encompass everything previously said, the following US are written:

- US7.3 – Fix buttons of Action Plan creation form.
- US15 – Permission system

Please find their full description in the Annex section Iteration 4 - New User Stories.

5.6. Iteration 5

In this iteration we will work in the following User Stories:

- US7.3 – Fix buttons of Action Plan creation form.
- US10 – General view for guides and action plans to offer at least CRUD functionality.
- US15 - Permission system.

In summary, we will first modify the creation form buttons to match the client's requests. Then, we will investigate and develop a permissions system, defining a rule set about how the content will be shared throughout the system. Lastly, we will expand our Creation Platform by developing general views for Plans and Guides from which we will be able to manage all the content.

Now we will divide each of these stories into smaller tasks and write the corresponding acceptance tests. Please find all of this in the Annex section: Iteration 5 - Tasks and Acceptance Tests.

5.6.1. Development

We decided to start with US7.3 since it will take us no time to develop. This is the result:

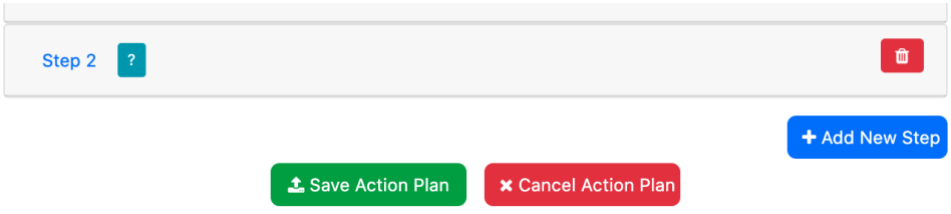


Figure 48 - Create Action Plan buttons - Iteration 5

Our content Creation Platform is growing considerably, and we are not structuring the code in the best way possible since we are only using PHP files with the HTML code inside it, mixing the visual side of our website with the data processing and backend logic. For this reason, we decided to refactor our code to follow the software design pattern: Model-View-Controller, usually known as MVC. This pattern aims to separate the program logic (Controller) and the data (Model) from how it is presented to the user (View). To apply this, we can separate the files into two new ones: the PHP in one side, with all the program logic and data managing, and on the other, HTML files that will be in charge of representing the information. With this change, our code is more modular and as we are separating two different concepts, its development and maintenance in the future will be simplified. From now on, we will develop all new scripts in this way.

As US10 depends on what is decided and developed on US15, we will develop this last one next. We want to offer creators the possibility of choosing whether they want their content to be used by everyone on the system or only by a small part of it. But what are the implications of doing so? To answer this question, we studied Actions Plans and Guides as separate scenarios and analyze how this change would affect each of them. We will start with a basic permission system: content can be or public or private, and we will expand from there if we need to.

Right now, creators can only interact with their own content, so the next logical step is to allow them to share their creations or at least give them more possibilities. In the case of Action Plans we would like to add two new features: if a creator likes an already created Plan from another user but would change something from it, we want them to have the possibility of creating a copy and modify it as they wish. Second, when creating guides, we want users to be able to choose from any created Action Plan, not only theirs. This means Action Plans' privacy only affects the Creation Platform and it is easy to implement: if the plan is public you agree to it being copied and selectable for anyone's guide; if it is private, only the creator can use it.

Then, we followed the same thought process but with Guides: we consider that allowing copies of other creator's Guides can be troublesome since it could include private Action Plans, which other creators should not have access to. We also thought about allowing copies of your own Guides but right now, this only means copying the title and the selection of plans, so we will not include this option either. With this, we noticed that Guide privacy did not affect the Creation Platform itself, but it did to another component of our architecture: the Visualizer. If a Guide is public, this could mean that all the consumers (Visualizer users) would have access to it, whilst if it was private, it would need to be accessed with another more secure mechanism.

It looks like we now have a better idea of what to do, but before doing any changes to the Creation Platform we would like to make a small spike to determine how we can identify consumers in the Visualizer. This is because if we, in some way, want to offer private Guides in the mobile app, we will need to keep a register of who is really having access to what content.

The first option we considered was to implement a register system as we did in the Creation Platform, but this would certainly be tedious for the users and does not make much sense due to the nature of our application. Another option could be to try to get the Gmail account associated to each android phone, but after some further investigation in how to do this, it required several user permissions and implied many workarounds to retrieve it from code. So, we searched for a solution that did not require much from the user but through which we would still be able to identify them. After some research we found a very thorough report by Sylvain Saurel [36] about the different possible ways to identify an Android device, which was exactly what we needed.

In it, we learnt that there were some possibilities like retrieving the Unique Telephony Number (UTN), its MAC address or its serial number but possibly the best solution out of all of was to get its UUID, which identifies a specific installation and not the physical device itself. The advantage is that it does not require any other peripheral like a SIM card (UTN) or relies on other conditions as the Wi-Fi being enabled (MAC). It is also quite flexible, since if used in conjunction with the Android Backup service, we could still identify it when installed in another device. The report even included a piece of code to demonstrate how to obtain it, presented by Reto Meier in a Google I/O presentation. With this we have all the information we need, so let's make it work.

We created two new tables CONSUMER and CONSUMER_GUIDES. The first will store the UUID to identify the consumer and the second one will hold which guides that consumer has access to:

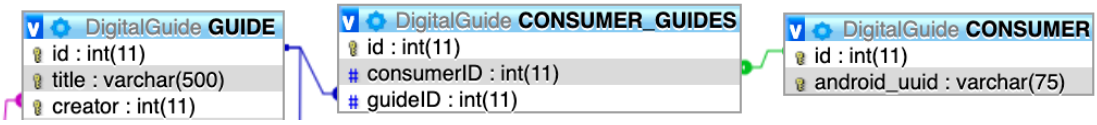


Figure 49 - CONSUMER and CONSUMER_GUIDES tables

Then we created a small API called *createConsumer* that is in charge of inserting new consumers. In Android Studio we update the functionality behind the splash screen to get the UUID and call this API in order to complete the register. This worked perfectly, so the spike is completed.

We are ready to go back to where we left off and start with the renovation of the different components to develop the permission system. As always, the first thing to do is update the database, where we create a new table VISIBILITY and insert two rows one for public and the other for private. We also add new columns for PLAN and GUIDE to reference it.

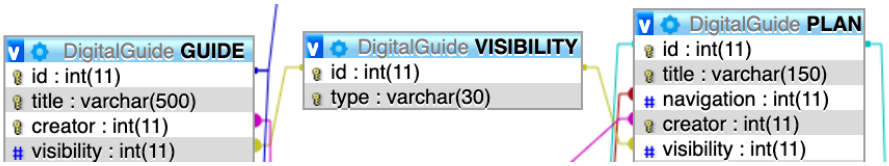


Figure 50 - VISIBILITY table

We have a clear vision of how public and private Action Plans are going to change our system, but we are not as sure about private Guides: we have to create some mechanism where

consumers can gain access to private Guides, but how? The best option we may think of right now is that when creating a private Guide, a random code associated to it is generated, which can then be shared by the creator. To access it in the Visualizer, we can have a new Activity that lets you insert codes and get the corresponding guides. It is not a foolproof method, but for the moment it may work for us. We then update the creation forms to show a modal after clicking the save button, which will tell creators the implications of making the content public or private:

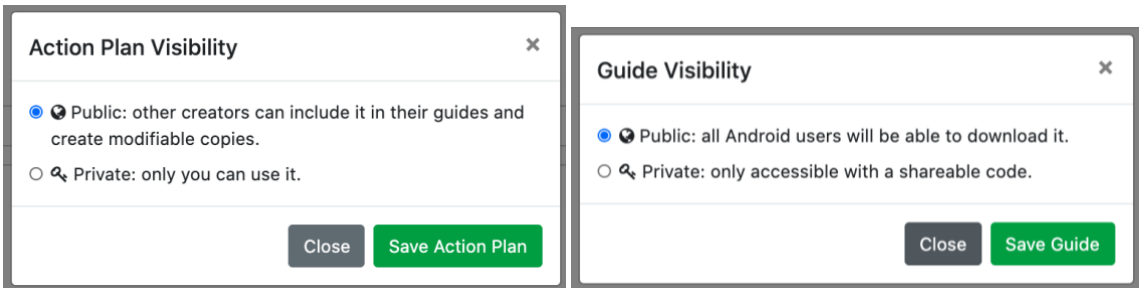


Figure 51 - Visibility modals for Action Plans and Guides

And of course, we update the submission scripts to also save this information. We wanted the random code generated for private guides to be somewhat secure so went to *Stack Overflow* to find what forums had to say. The code we found may be a bit too overkill for our needs, but it is a secure, random string generator made by Scott Arciszewski [37] which is even customizable: we can define the key space the code is generated from and how many characters long it will be. We decided our key space to be all capital letters (A-Z), numbers (0-9) and six characters long.

To test it later, we create a private guide, noted down its code (S7QVS7) and headed back to Android Studio to start with the upgrade. Earlier, we downloaded all guides and plans from the first moment, but now that we are developing a new version, we should improve this. What we will do is only download the public guides from the start, which then must be selected from a new activity where consumers can choose which guides they want in their devices.

In addition, plans will not be downloaded from the start either, since we do not know which Guides the user will want in their device, so we will fetch them when a Guide is going to be viewed. Let's make these changes and develop all the necessary updates to support public and private content. After several working hours, this is what we came up with:

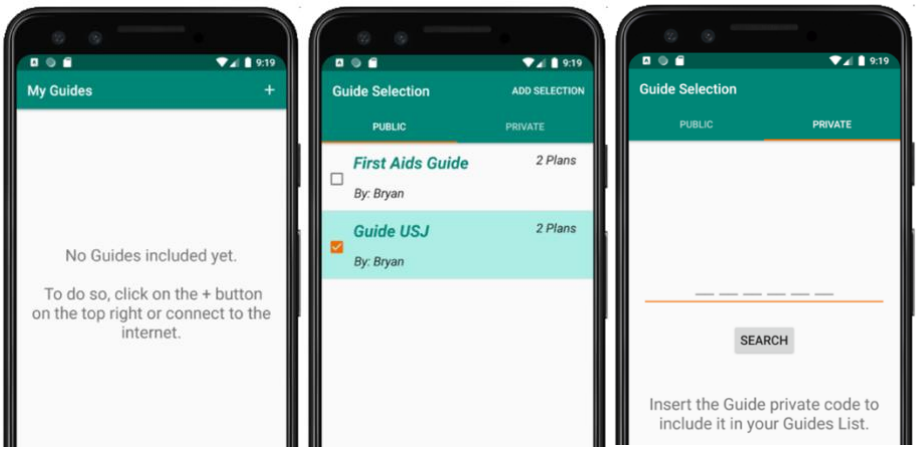


Figure 52 - Public and private guide selection in Visualizer

The leftmost picture is the first view the user will see after the splash screen, where no guides have been selected yet, but this can be done by clicking the "+" icon on the top right. This opens a new tabbed Activity, which includes tabs at the top to change between looking for public or private guides. Remember the fragments we talked about in iteration 3? Our investigation there was not in vain, since we are now using them to show what is inside of each of the tabs, which means that when the other tab is selected or swiped, we change the fragment we display. As we can see, both have very different views: in public, we show a list of all the public guides available, which the consumer can then select and add to their phone by clicking the "ADD SELECTION" button on the top right. On the other hand, in private, we provide the means to write the private code of the guide users want to have access to. When clicking the "SEARCH" button, we first make sure that the input received by the user is valid and then we make a call to a new API that will search for that guide and if it exists, allow the user to add it to their guide list. If the same code or the one of an inexistent guide is provided, we display the appropriate alert:

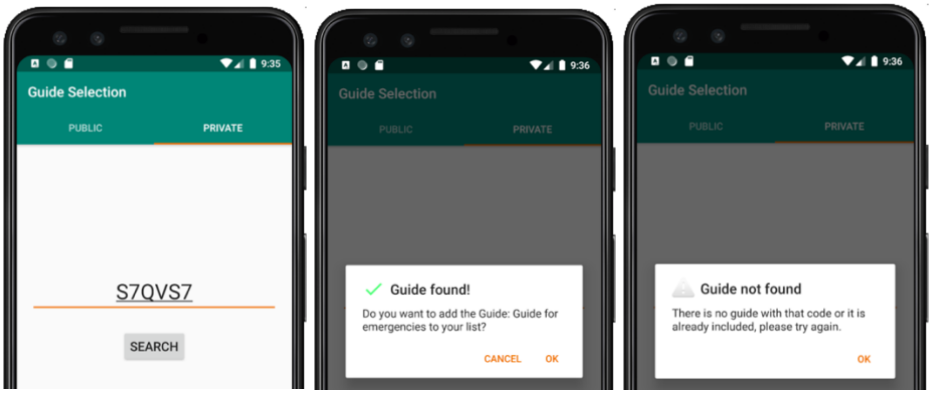


Figure 53 - Fetching private guides in Visualizer

When the consumer chooses to add certain guides, we also are updating the CONSUMER_GUIDE table in order to remember their selection when the app is launched at a later moment. As we were at it, we also added the possibility to remove guides from our list, and this can be simply done by swiping the row the guide is in to the left. When this is done, we also remove the

corresponding entry from the database. This works but could be improved in terms of UX, since we do not show any kind of indication that you are deleting it: a red color indication paired with a trash can icon when swiping for example, could fix this problem.

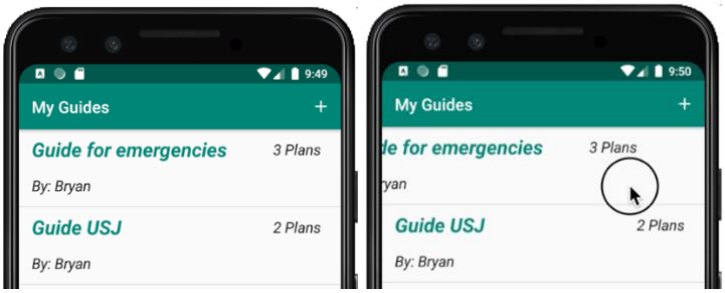


Figure 54 - Guide list view and guide deletion - Iteration 5

This is all we can develop for US15 right now so let's move onto US10. We want to create a default view both for Action Plans and Guides where we are able to manage them easily. The main information we want to see about plans is their name, who created it, its visibility and then, for each of them, we want the options of edit, duplicate and delete. It looks like there are many things we want to display, so the first option that comes to our mind is to create a table with all the data, and the options could just be buttons on a separate column. To this, we also want to add plans that are made by other creators, but the actions should change, only allowing duplication. When developing this, we also noticed that right now, as there are not many plans, it is really easy to find the one you are looking for. Nevertheless, in the future this table could grow exponentially, so we should look for ways this could be improved but for now, our temporal solution is to implement a search bar that filters the rows by the plan's name.

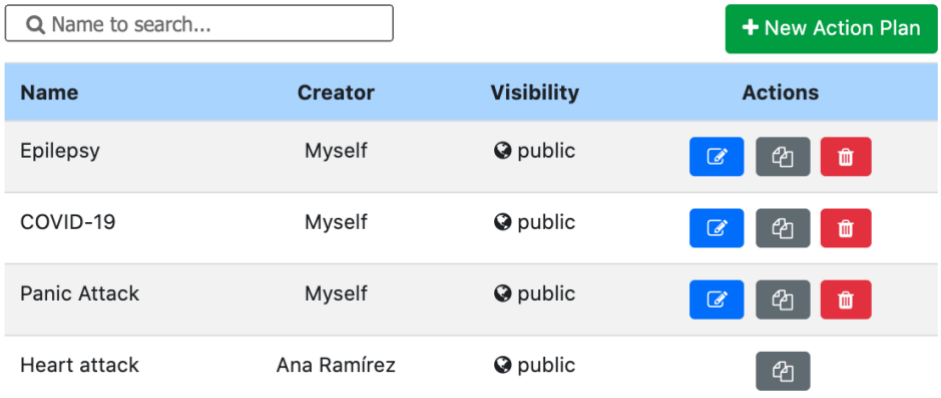


Figure 55 - Action Plans main view

Now we just have to implement the different actions and probably the most challenging one is editing. In this option we want to show all the information of the selected plan but in a view where it can be modified. At first, we thought about a simpler view than the creation one, but we soon realized that if we wanted to offer full edition of the plan this could not be done in another

way: we have to create the same form as in creation but with all the information already filled, so that creators can see what was previously there and modify it easily.

The way we solved this was to have a base HTML that works both for create and edit, but the file that contains it is actually a PHP and not a HTML. This allows us to break the HTML and insert PHP code where we need it to, allowing different functionalities for creation or edition. In the first case, we just made the necessary changes so that it worked exactly as before and in the second, we inserted the code in charge of filling all the inputs of the form dynamically.

Now the form is completely filled and ready to be edited, but uploading it will have its complications: when filling the form, we are not inserting the image again but just writing the previous file name for the creator to know which file they uploaded, so input is actually empty; steps can be deleted and new ones added so it is not as easy as just updating the previous steps. The rest of the logic will probably be the same as for creation, but we will not be sure about it until we develop it, so let's do it.

Indeed, uploading edited plans was more complicated but we managed to do it. Let's comment the main differences: we had to distinguish in some way the already existing steps from the newly added ones, so what we did was modify the base HTML and created a new hidden input in all step cards, whose value would be zero by default. Then, when we filled all the inputs with PHP, we change this value to the corresponding step ID. This brought several advantages in the POST script: when we gathered all the inputs from $_POST, first, we compiled all the step IDs and insert them into an array. Then we processed each step, if its previous ID was zero, it meant it was newly added, so we execute an INSERT statement and follow the same process as in create, meanwhile if it was different from zero, it meant that this was a previous step which was now modified, so we need to execute an UPDATE query. For this last case, we also removed the ID from the array, in this way when we finished uploading all the steps, the IDs that remained corresponded to deleted steps, so we could proceed to remove them from the database.

On the other side, this is how we solved the problem with media files: when there is a step that contains files, there are two possible case scenarios: the creator changes it or not. As we have mentioned previously, our form uses frontend validation to make sure all inputs are filled before submitting; this is a problem, since our file inputs are actually empty, so if creators want to keep the previous file, they would not be able to. To solve this, when filling the form, we remove the *required* condition from these inputs, which will pass the validation when submitting, but when looking to retrieve it from $_FILES, there will be nothing to retrieve. Nevertheless, as we know which step type they have chosen, if we then seek for the corresponding input in $_FILES and there is none, we internally know they have chosen to keep the previous one.

For the other case, they are providing a new file, which means we have to first check its validity. When verified, we could proceed to save it in our storage, but we have a problem: we always prepend the step ID before the file name, this means we would have two files with the same ID,

and this could be an issue when the API tries to retrieve it for the Visualizer. Fortunately, this is an easy fix, delete the previous file first and then store the new one. Lastly, we can update the step entry in the database to save the new file name.

We have made possible the edition of plans but there is one final touch we would like to do to improve its security: the plan to edit is determined by parameter *id* in the URL, this means it is exposed to the user and therefore they could access and edit any existing plan by changing this number, even the ones belonging to other creators. To avoid this, before loading the filled form, we make a small query that checks that the plan exists and belongs to the creator who has opened it. If these conditions are not met, we redirect them back to the Action Plan's main view.

The two remaining actions to implement are duplicating and deleting plans. For both of them, we need to update the main table to apply the changes, but we do not want to reload the entire webpage to do so. Therefore, we decided to use AJAX, which will allow us to fetch whatever PHP code we need in the background and show the results to the user as they are ready.

Deleting is straight: we just need to delete the corresponding entries from the tables PLAN, PLAN_STEP and STEP. When we receive a success response from AJAX, we just need to remove the row from the table. Duplication on the other side is a bit more complex: we have to make an exact copy of the plan, which does not only imply the entries for the tables PLAN, PLAN_STEP and STEP, but also the actual files that the steps include. This was not very complicated to do, since PHP provides all the necessary functions we needed. However, it is interesting to mention that when we updated the uploading scripts to support creators, we also established the condition that no user could create two plans with the same name since it could cause some confusion. To avoid falling into this error condition when duplicating, we establish that the new plan will be called as the previous but with the word "Copy" appended at the end. This means that if we duplicate a plan called "Epilepsy" the result will be "Epilepsy Copy". This is not the best solution, since if we decided to make a copy of "Epilepsy" again, the duplication will not succeed since "Epilepsy Copy" already exists. Our priority was for it to work, so we will fix this in the future.

Now we will do the same thing but with Guides. The information to display in this case is: title, number of Action Plans it includes, visibility, the code (if private) and as actions, edit and delete.

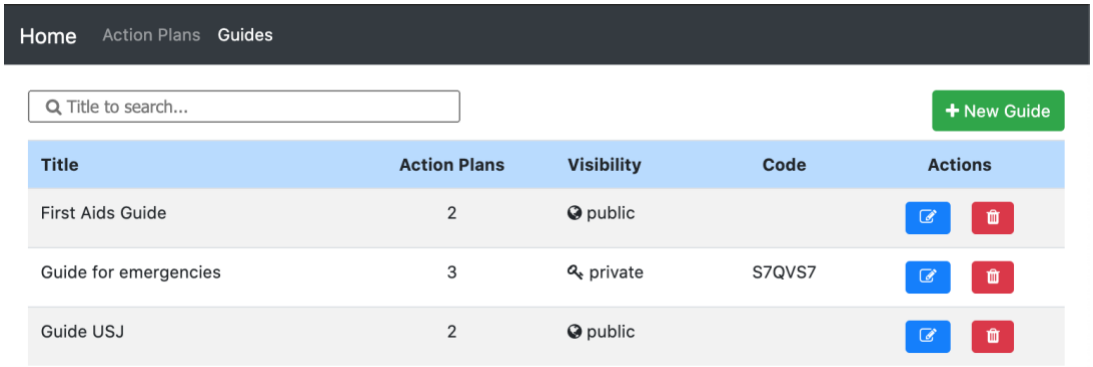


Figure 56 - Guides main view and navigation bar - Iteration 5

In all this time we have not updated the Guide creation form, which means it needs a serious rework to make it look and feel the same as Action Plans. Nevertheless, there is a User Story we will like to work on next about Guide customization. As the creation form is going to change, it does not make sense to implement edition right now, so we will do it when the renovation is ready. Nevertheless, the deletion can be done without problem, so we developed that.

With this done, we noticed that we should update our navigation bar and the home screen, since it makes more sense to redirect creators to these main views that to the creation sites as we did before. Now the navigation bar just has three elements: Home, Action Plans and Guides, and we highlight with a brighter white the one the user is right now, as we can see in Figure 60.

With this, we finished the tasks required for this iteration, estimated in 30 hours of work, but as it has happened to us before, we had to face some complications and further investigation to complete it. This resulted, in eight more hours than expected, making a total of 38 hours of work.

5.6.2. Meeting with the client

After presenting the work done to the client, we run the acceptance tests to verify we have accomplished all our objectives. All of the tests passed except two, which were the ones related to Edit Guides that could not be done in this iteration:

- Create a new Guide, select one plan and save it. Edit this Guide, change the name to My guide duplicate and select two plans now. Check that it has been updated correctly.
- Edit it again and uncheck the selected plans. Frontend validation should not allow us to submit it.

Having reviewed everything, the following observations were made by the client:

- Editing Guides is a must, we should implement it once the renovation is done.
- Duplicating Action Plans is a good way of allowing other creators to modify the already created content but duplicating the files in our filesystem is very inefficient since we are storing the exact same files twice. We should implement the necessary changes to avoid this but will also like to improve upon it: when selecting a step that includes files, we would like to have a file selector that will show all the already uploaded files to the system, which the creator can then choose or upload a new one as they did before. We can even go one step further and also introduce our permission system where files can be marked as public/private so that other creators are also able to use them.
- As already mentioned, the main views for Action Plans and Guides are quite useful but could also grow in size very quickly, especially if the project was released to the public. We should investigate further ways we could limit this table's size, like include pagination so that only a set of rows is showed at a time, or more advanced filters like: show only our or others content, by number of plans it includes (in the case of guides), etc.
- At some moment we would like to make the platform available to real users and to do this, we should move the Creation Platform from our local machine into a real server.

To encompass everything previously said, the following US are written:

- US10.1 – Edit Guides.
- US16 – Improved files management.
- US17 – Guides and Action Plans tables size control.
- US18 – Move Creation Platform to real server.

Please find their full description in the Annex section Iteration 5 - New User Stories.

5.7. Iteration 6

In this iteration we will work in the following User Stories:

- US6 – Improve Guide creation to allow personalization.
- US10.1 – Edit Guides.
- US12 – Caching in Android app.
- US18 – Move Creation Platform to real server.

In summary, what we want to achieve is: offer guide customization, so that each Guide can have its unique look and feel in the Visualizer. When done, we will be able to implement Guide edition from the previous iteration. Next, we will develop a cache system in the Visualizer to avoid fetching all the content each time we load the application, looking at the same time into how we can make it work without internet connection. Finally, we will move the Creation Platform to a real server so that it can be accessed by everyone from anywhere.

Now we will divide each of these stories into smaller tasks and write the corresponding acceptance tests. Please find all of this in the Annex section: Iteration 6 - Tasks and Acceptance Tests.

5.7.1. Development

To start with US6, first we must think of what we can offer: the client suggested that it would be interesting to allow some sort of branding image or logo that can make guides stand out from the rest. As another idea, we could also ask for a color to use in the Visualizer to provide a different aesthetic to the content. We will try with these options right now but leaving room for improvements in the future. But first, let's renovate the outdated Guide creation form:

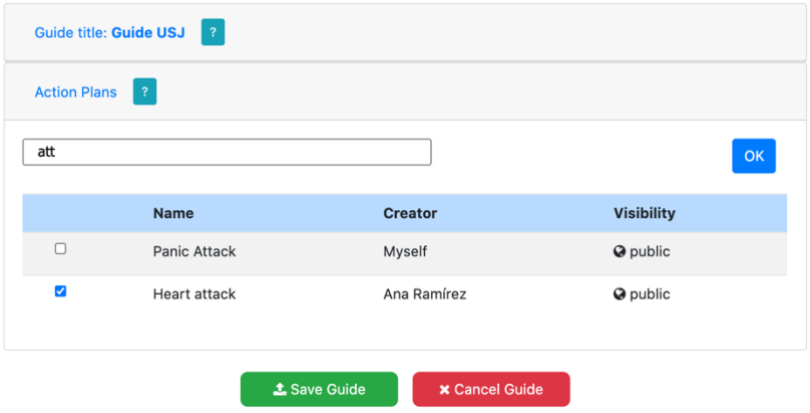


Figure 57 - Guide creation form - Iteration 6

As we can see above, each of the form sections is now inside a card and everything works as with plans. We also adapted the table we built for the Action Plans main view with its search bar, since it will help creators to choose which plans they want to include in their guides. Next, we add a new card with the customization options we want to offer: a file and a color input.

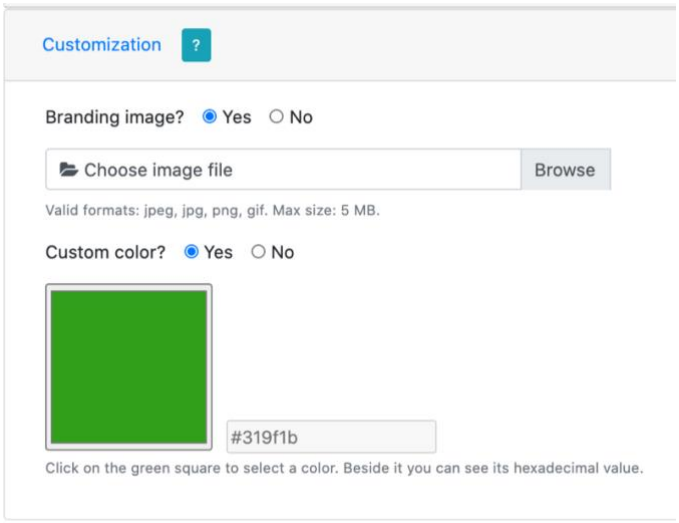


Figure 58 - Guide customization form section

We noticed that in some cases creators may not want to provide a branding image or a color to personalize it, so we give the option of completely avoiding them if desired. In order to store this new information, we update the database by adding to the table GUIDE two new columns: *branding* and *color*. In relation to the POST script, we decided we would store the branding images in the same folder we have our step files in. This means that if we used the same system of prepending the ID (in this case of the guide) to the file name, it could crash with some already existing file and would cause some problems later. For this reason, we applied the simple solution of adding the letter "G" before the ID, resulting in a name like: *G1-USJ.png*. This could be done in other ways, such as having a separate folder for these images, but this solution works for now.

Now in Android Studio, let's investigate how we can apply these customization options. For the branding image we just need to find out where it fits better and after trying some different options, we found that it made more sense to show it in those activities where Guides are listed. We tried different layouts to see where it fitted best and chose the following one:

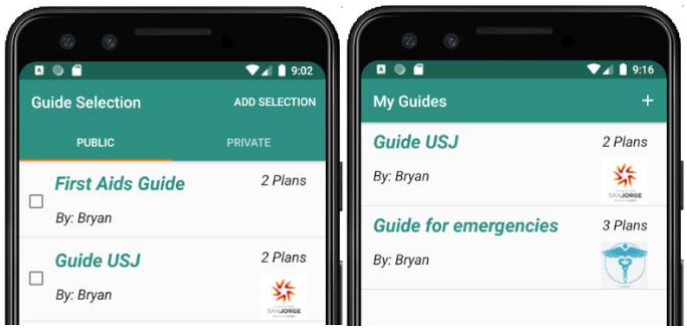


Figure 59 - Guide branding

On the other hand, allowing color personalization is not a trivial task since we need to have many factors into consideration: if we paint texts and the background is always white for example, some colors may not be visible; if we paint the background instead and the text color is fixed, we may have the same problem; if we use too many colors (each guide has its own) it may end up being too chaotic; if we then consider visual disorders our users may have such as color blindness or even epilepsy the problem escalates exponentially. As there is no perfect solution for this, let's try different options and see what works best.

The elements we can use color on are: toolbar, titles, background and buttons. We think that coloring the toolbar is a good option but not in all activities since some of them are Guide lists, meaning there is not one color to choose from. Nevertheless, all the content inside the Guide (plans and steps) could use this color. Although this is a good option, we still have to solve the problem of what text color to use on the toolbar, since we want it to always be visible. Fortunately, we found a solution provided by *StackOverflow's* user *SudoPlz* [38] that given a background color, it returns white or black depending on which one will be seen better on top of it. We can also use this result to paint the directional arrows inside the Buttons layout so that it matches the toolbar.

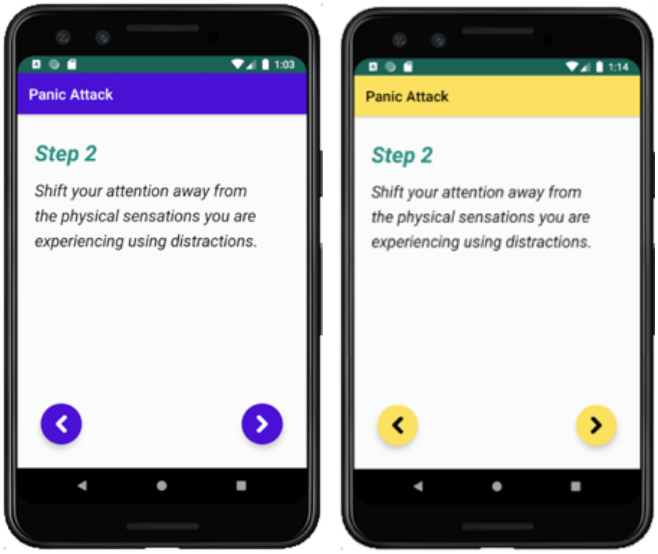


Figure 60 - Color customization with white or black accents

This looks reasonable to do but it certainly would look better if the title also matched this color. Here, we have to make a decision: if we keep the background color white, we have to find a way to make the titles always readable, meanwhile if we change the background color, the title color would need to be adapted. Due to the nature of the application, we decided to keep the white background color ensuring it will always be readable. This means we have to find a solution to ensure titles will always be visible, independently of the color the creator chooses.

Our first option was to draw a dark shadow around the title so that it would always make contrast with the white background and therefore light colors could be seen without problem. After some trials we found out that many colors caused a sort of blur effect which all in all, made it even less

readable, so we discarded this option. We also tried painting only the title's background, but it had inconsistent results and did not look good most of the times. After some investigation, we found a possible solution from this *StackOverflow* thread [39]: we could use a function that from one color, it returned lighter and darker versions of it, making a sort of palette. So, for example, we could use the darker version for the title color, where readability is important, and the other colors for elements where this was not as relevant or could be solved in another way.

We adapted that solution and after some trials we got the title to always be in a color suitable for the white background which is great. The only inconsistent part of this solution is that when the creator chooses a very white-like or light color, the resulting one ends up being a greyish tone, which may confuse some users, but it is a small sacrifice we can assume in order to have the expected results.

With this done, plans and steps have their customization, but we would also like to give some color to the activities where the Guides are listed, so we tested some alternatives: with a palette of two colors, A) Use the light color for background and dark for the title. B) Do the same but inverting them. C) Use the dark color for background and white or black title based on it. D) Only use dark color on the title. E) Alternative version of A but with more tone difference.

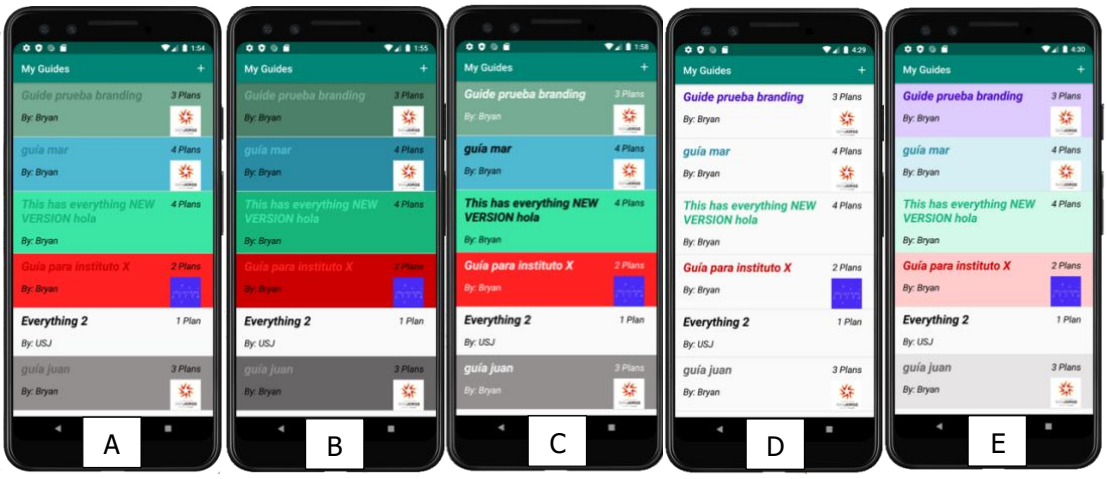


Figure 61 - Color trials

Our favorite results from this experiment were D and E since they are the most readable and do not look too chaotic, even when we are using very different colors. Between these two, although E makes a better use of the chosen color, we feel like D is the most reasonable choice since it looks clean and readable and still uses the color in a smart but subtle way. We also decided that when a Guide has no color customization, we will use black as the default color, which we will also use for the default UI elements to avoid mixing too many colors in the different views.

With this we have successfully finished US6, so we can proceed with US10.1 which is editing guides. Having done the editing for Action Plans this is quite a straightforward task since we just have to apply what we did before but now adapting it to the Guide form. We developed it all, including the POST script, tested that everything worked and marked this story as finished.

Now we are going to proceed with US12, one of the most relevant stories for the development of this project. The objective is to create a version control system to cache the content we have already downloaded, which will allow us to highly optimize the requests to the database and if done right, will allow us to use the application without internet connection.

We can divide the work to do into three different problems: the first is how to determine what needs to be downloaded again and once we know it, do we request the whole object (Guide or Action Plan) or the exact parts that need to be updated? Second, we have to investigate how to store the information inside the Android device; the IDE provides some solutions, but we must investigate which one works best for us. Last, make it work without internet connection.

We may start with the easiest part and this is determining which content has changed. Possible solutions consist in saving extra information which we can then use to compare and determine if they have been modified. For this, we could store the timestamp of when the content was uploaded or a simpler version of this where we associate a number to each upload, so when content is created, this is set to one and in further editions it will increase one by one. We decided to go for the simpler solution, and as we always create and edit Guides or Action Plans as a whole, it does not make sense to store the version for each of the inputs they are composed of, since the version is going to be the same for all. So, we created a new column *version* both in GUIDE and PLAN and updated the creation and editing scripts so that it is initially set to one and further editions increases it by one.

Now with the next problem: how we store this data in the mobile phone. We have three main options to go for: SharedPreferences (SP for short), files or SQLite. SP is an easy to access file that stores key-value pairs and which Android Studio provides simple methods to read and write from. Files on the other hand, work just like in any other device, we can have any number of files associated to our app where we can store the necessary data. Lastly, SQLite is a relational database management system stored directly in our mobile device.

In this case we made a small spike to understand how each of them worked and make a wiser decision about which option to go for. The difference between using SP or files was quite minimal, but SP was certainly easier to interact with. SQLite on the other hand will be quite cumbersome to use since we would practically need to replicate one to one our real database, so we would have to be very cautious of always making sure data is consistent throughout. In addition, extracting the necessary objects, manipulating them and updating them is certainly more complex than with the other alternatives, so we discard SQLite for the moment.

Back to the other two options, both of them are good choices and will work for our case, but given that SP is a bit more convenient, we will use it for now. Nevertheless, we must comment that it is discouraged to save too much data in SP files, which is not a problem for now since we are only storing plain text and not much of it either. If the content starts to grow considerably in the future, we would probably need to switch to using regular files or even consider SQLite again.

We will have three different SP files: Guides, Plans and GuideIDs. This last one will be used to have a register of which Guides the consumers have decided to have in their phones, information which will also be stored in the database to make it persistent. At initializing the Visualizer, we will consult this file to know which Guides to show in the main Activity. On the other side, this list can also be modified by the consumer by removing a Guide from the aforementioned Activity or new ones added through the Guide selection Activities. When this information changes, we must always keep the database updated.

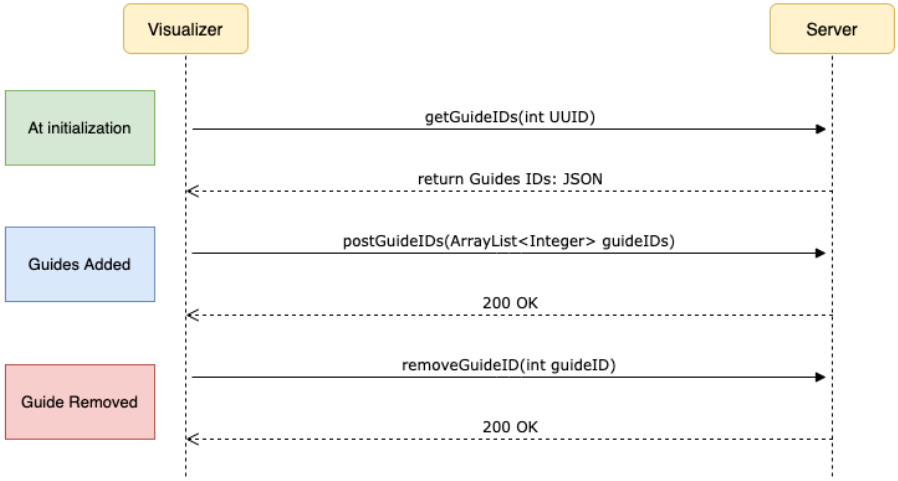


Figure 62 - GuideIDs sequence diagram

Now we get into the interesting part, we have to develop a system where we only download once all the content and store it in the device. From this moment onwards, we must request just the new or modified versions of the existent content. To help us in this process we create a new data structure that we are going to call Cache, which just contains two integers: one for the ID and the other for the version, which we can then use both for Guides and Action Plans. The first time we start the Visualizer, all public Guides are requested automatically; the user can then move on to use the app, for example requesting some private Guides, which we will store in the corresponding SP file.

The following times the application is opened, we will have to follow this process to make sure we only receive the content we do not already have: first, get the corresponding SP file of let's say Guides for example, process the JSON it contains by converting it into the analogous Java objects and insert them into an ArrayList. We will then iterate through it, creating a Cache object for each one and store them into another ArrayList. When this is done, we send this Cache list to the server, which will then be in charge of checking one by one if the duple id-version that we sent corresponds to the one in the database. If they are different this means that the content has been modified, so we fetch that Guide and add it to the response array. In the process, we are keeping the register of which content has been requested, which means that if at the end, there are Guides that have not been checked, they must be new content that the user does not have, so we also add them to the response.

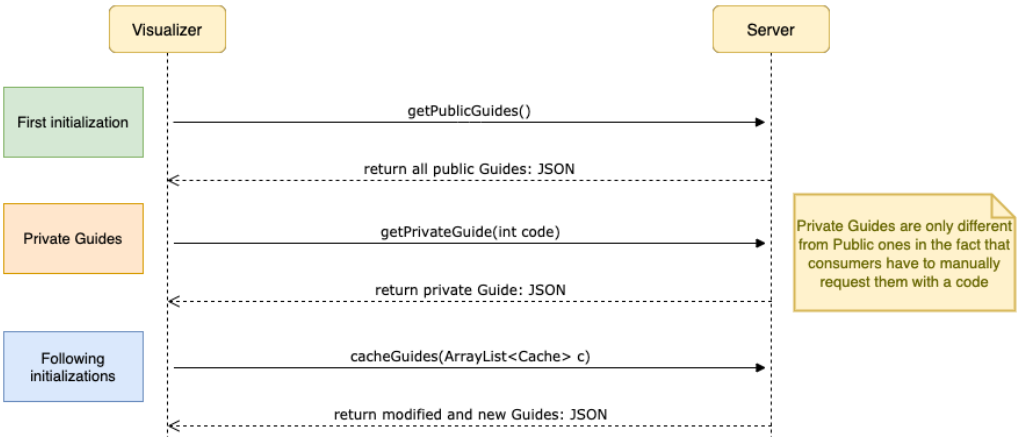


Figure 63 - Guides sequence diagram

When receiving this data, we follow this other process: if the response is not empty, we must convert the returned JSON into Java objects we can work with. We still have access to the ArrayList of the stored Guides, so what we need to do now is iterate this list looking for the Guides there is a new version of and replace them. Then we also add those Guides which we did not have a previous copy of and with this done we have an updated ArrayList with all the latest content, so we just need to convert it back to JSON and overwrite the SP file with this new data. Lastly, we update our Singleton to be ready to work with the latest information.

These processes work in the exact same way with Action Plans so after making the necessary changes to make it all work, we have managed to store most of the content we need in our device, with the exception of media files which we have not talked about. Glide caches images in the device by default, so this is done automatically for us. Videos on the other side, will not be as easy to store and manage, and as it is quite critical to do it in the best way possible (since they can consume lots of the device's memory and we should avoid making the app too heavy), we estimated that we would not have the time in this iteration to find and develop a good enough solution, so we took the decision of leaving it as future work.

Now we are able to tackle the last problem of this story and it is making the Visualizer work without internet. The first thing to do is simple, we must find out how to determine if the device has internet connection or not, and searching online, we quickly found this solution [40] that exactly returned this information. That is to say, that having an active network connection does not mean that there are no network issues, bad signal, server downtimes, etc. but as that would not be as simple to check, we will stick to this solution for now.

Thanks to our previous work, allowing this change is quite simple: if there is no internet, we just get the SP files, retrieve their data, convert it and then just display it as we have always done. To avoid problems when the application comes back online, we applied certain restrictions to the offline mode such as: not allowing the removal of Guides or selection of new ones, since if you have not downloaded the plans associated to them, you would just have access to the basic Guide

information (title, author, version and number of plans) which is essentially useless. With this we just needed to cover all the extreme cases where errors could happen, as what to do when the SP files are empty or when some content is missing. Then, we just made some tests to be sure the result was the expected one and we concluded that we were ready to continue.

The last User Story we will work on is US18, where the objective is to move the content Creation Platform from our local machine to a real server. The first decision to make is which hosting service we are going to use and of course, we will go with the free route. We really liked the advantages *InfinityFree* offered, so we tried with it first, but, when we migrated our database, the free hosting one did not support the storage engine InnoDB but only myISAM. This ultimately meant that we were not able to have foreign keys and as we already know, they are pretty much crucial for how we structured our data. As this option did not work, we changed to *000Webhost* which we had already used previously and knew it worked.

We set everything up with this service provider and after some testing it seemed that everything works fine but actually an important part did not. As we may recall, in the registration phase, we sent an email to the creator, so we could check it was a valid email. It turns out that sending this email from our local machine did not cause any problem, but when sending the same one from the online platform, it was marked as spam. This is because we are sending a URL inside the email, which usually has some malicious connotations, so Google directly blocks them.

We tried to solve this by creating a new email account or sending the URL in different formats to see if we managed to pass through but, of course, Google's spam system is much more intelligent. Gmail uses the OAuth 2.0 protocol for authenticating a Google account and authorizing access to user data, so if we wanted to maintain our platform as it is, we would have to gain this authorization. The process of obtaining the credentials and access token was quite complex, so we tried following some tutorials [41] [42] but after spending too much time on it, we could not manage to continue, since we kept receiving the following error in one of its steps:

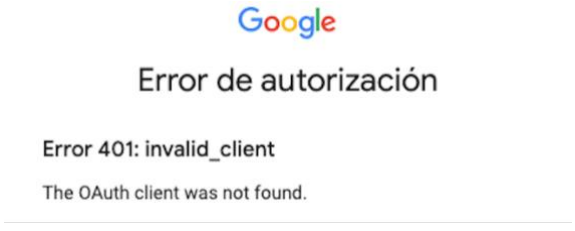


Figure 64 - OAuth 2.0 Authorization error

As we did not manage to follow this route, we had to adapt and find another solution to the problem. As we were not able to send the URL, we ended up just sending a code, which afterwards the user should copy and paste it onto the website. Obviously, it is not the cleanest or most secure solution and definitely, should be improved in the future but at least, it works as a temporary fix to the problem.

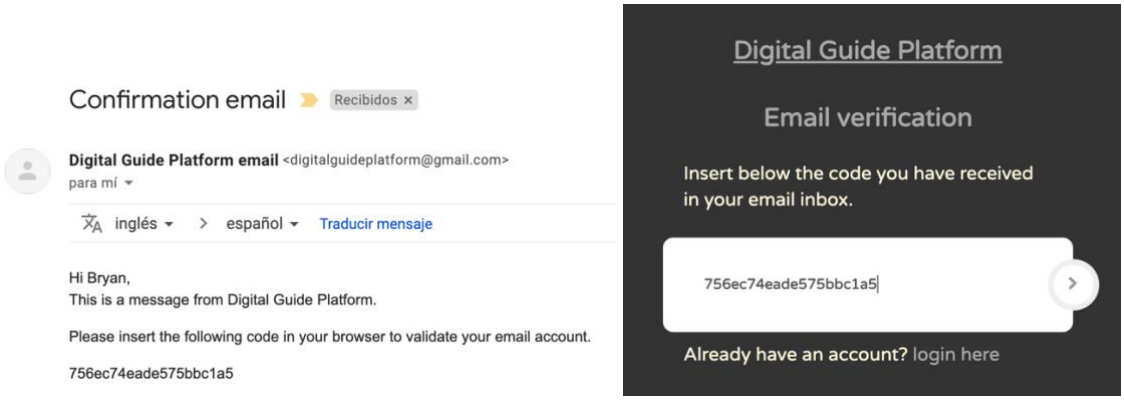


Figure 65 - Email with code (left) and new screen to insert it (right)

Just as final touches, now that we have moved onto "production", we also updated all the placeholders we used in the creation forms with new more up to date versions that correspond better to reality, making sure that creators will have as little doubts as possible when creating their content.

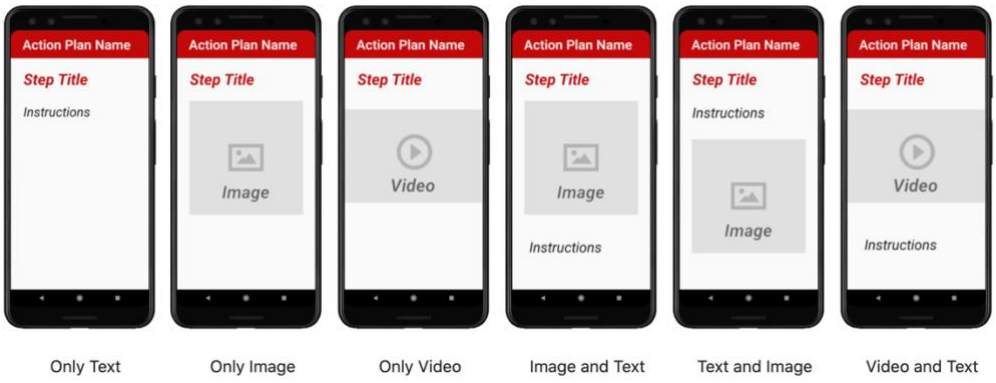


Figure 66 - Step type options - Iteration 6

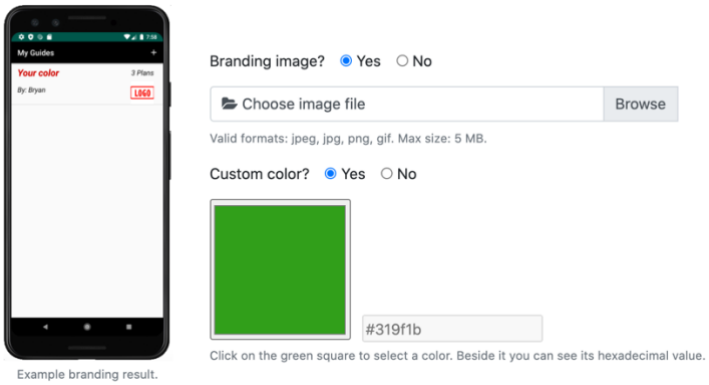


Figure 67 - Guide customization example

After all this hard work, our platform is finally online and operative, which everyone can now access through the link: http://digitalguideplatform.000webhostapp.com/. And with this, we can mark both this story as well as the whole iteration as finished.

As it has happened with all previous iterations, our estimation of 30 hours certainly fell short, taking us ten more, investing this extra time in solving problems we did not expect such as making color customization work properly, emails being marked as spam or that extra details we included at the end to make it feel like a more finished and complete product, making a total of 40 hours.

5.7.2. Meeting with the client

After showing the client the work made, we run the acceptance tests to verify we have accomplished all our objectives. All of the tests passed except one:

- Close the Visualizer and turn Airplane mode on (no internet). Open the Visualizer again and check that all the content we had already visualized can be accessed and works as expected, including all media types.

Practically all of the content works without internet connection with the exception of videos, which we did not have the time to develop a smart system to store them efficiently and access them whenever we needed it to, so we had to leave this functionality as future work.

Having reviewed everything, the following observations were made by the client:

- As we also mentioned, the client would like to see an improved version of the caching system, which could be enhanced in two main ways: the first is based on developing a more intelligent system where we do not download the entire Guide or Action Plan when it is modified, but only the exact parts that have changed. Secondly, it should not be as rigid as having internet or not; we could implement a system that checks the quality of your connection and waits if it is not good at the moment or just ask the consumer whether they want to download them now or later.

Aside from this, the client is very satisfied with all the work done and as the objectives and requirements have been fulfilled, we can consider this to be the final first version of the project.

To encompass everything previously said, the following US are written:

- US12.1 – Advanced caching in Android app.

Please find their full description in the Annex section Iteration 6 - New User Stories.

6. Chapter 6: Economic Study

In this chapter we are going to calculate the real-life cost equivalent of developing this project, taking into account several aspects as the human resources needed, infrastructure costs, tools and resources used, and maintenance costs if it is continued.

This project has been entirely done by one person but in a real case scenario, it would have probably been separated into five different profiles: three developers for frontend, backend and Android, a QA/tester and a Project Manager (PM). Next, we will find what is the average salary for each of these positions in Spain according to the online platform *Glassdoor* [43]. These values correspond to the net salary earnt by the worker, but the company must pay approximately 30% more [44] to cover the social security expenses, so we will also calculate that corresponding amount.

With all of this, we will be able to estimate their salary per hour (considering there are about 1780 working hours in a year) and finally determine the exact cost of this project. As a note, this project has taken 420 hours to develop but they have not been distributed equally between the different roles: the majority of the time corresponds to the developers, who did have approximately the same workload; between tester and PM, we consider this last one to have a little more weight in the development (about 1/10 of the total time).

Profile	Annual Salary	With SS included (+30%)	Cost per hour	Hours worked	Cost for hours worked
Frontend	31.505 € [45]	40.956,5 €	23 €/h	115 h	2645 €
Backend	33.733 € [46]	43.852,9 €	24,6 €/h	115 h	2829 €
Android	33.065 € [47]	42.984,5 €	24,1 €/h	115 h	2771,5 €
QA/Tester	32.678 € [48]	42.481,4 €	23,9 €/h	35 h	836,5 €
PM	40.875 € [49]	53.137,5 €	29,9 €/h	40 h	1196 €
			TOTAL	420 h	10.278 €

Table 7 – Human Resources costs

For the development of this project we have used a 15-inch 2015 MacBook Pro laptop with a 2.2 GHz Intel Core i7 four cores processor, 16 GB of DDR3 RAM, and an Intel Iris Pro graphics, with a total value of 2200 €. This model has an estimated lifetime of five to seven years, so we will use an approximation of six years amortization period in the calculations. For testing purposes, we also used a Samsung Galaxy S9 phone, valued in 849 € with an average life of two years.

Concept	Annual Cost	Monthly Cost
MacBook Pro	366,67 €	30,56 €
Samsung Galaxy S9	424,50 €	35,38 €
TOTAL	791,17 €	65,94 €

Table 8 - Resources costs

Then, we are going to suppose we would work from an office, which in Zaragoza could cost in average from 250 to 400 euros depending on the location and characteristics so let's set it to about 325 €. Then we must also contract an internet service, but we do not require any out of the ordinary speeds, so we could use a cheap option as the one offered by *Jazztel* of only 28,95€ for 100 Mb fiber [50], more than enough for our needs. Then we must also consider some basic monthly expenses such as electricity, heating, water, garbage consumptions (100 €/month), cleaning (50 €/month), and other variable expenses (100 €/month), which all makes a total of 603,95 € per month.

In relation to the tools used during development (enumerated in section 5.1.1), all of them are free to work with, even for commercial solutions, so there will be no additional cost derived from them.

Now that we have a better approximation of the individual costs of developing this project in a real environment, lets sum them all up, taking into consideration that it has taken approximately 420 hours to develop. A month of full-time work is equivalent to about 160 hours, so we could say it took about 2,6 months to develop.

Concept	Total Cost
Human Resources	10.278 €
Material Resources	65,94 * 2,6 = 171,4 €
Infrastructure	603,95 * 2,6 = 1570,3 €
TOTAL	12019,7 €

Table 9 - Total project costs

Once the project is released to the public, the server still needs to be maintained and probably, *000Webhost's* free version would not suit as anymore, as it is limited to 300 MB of disk space and to a monthly bandwidth of 3000 MB, which would be easily consumed just by uploading a few videos to the platform. If we wanted to stay with this hosting service, we would have several options to choose from depending on our needs, but the most reasonable option would be to upgrade to a premium or business hosting account (which greatly improves the free-version characteristics) and which would cost us from about 4.79 to 5.59 €/month respectively for a 1-year plan [51]. We could also investigate other solutions, but for our needs we would not need nothing more expensive than 10 €/month.

7. Chapter 7: Results

As a small review of what has been done in this project, we first invested some time researching and investigating how traditional emergency guides had been converted into the digital world, finding some really interesting solutions on the way, but nothing similar to what we had planned for this project. We did not want to limit ourselves to just adapt one single guide to the mobile format but decided to go one step further and build from the grounds up an entire system that would not allow just the conversion of this guide but practically, the conversion of any guide.

After an initial phase focused on its design and architecture, during six XP iterations we gradually assembled the pieces to create a complete system. At the start it just consisted of a few web forms and a simple Android application, but at the end they transformed into a full-blown Creation Platform, which not only allows us to create Action Plans to describe any emergency procedure and Guides to collect them in a single place but also, offer their customization in the Visualizer, an application which allows you to view all the created content and even adapt it to your needs, deciding what Guides you want to have access and check them at any time, even without internet connection.

This of course is only the product of many hours of hard work and also thanks to the continuous communication and support of the client, responsible of making difficult decisions during the development but that ultimately greatly helped in the orientation and final result of the project. Nevertheless, all of it was not ideal, and as we will see in the following figure, we totally underestimated the time needed to develop each of the iterations but thanks to the decision of using Agile methodologies, we could totally make it work.

	Initial estimation	Final
Iteration 0	0	31
Iteration 1	30	35
Iteration 2	30	35
Iteration 3	30	40
Iteration 4	30	45
Iteration 5	30	38
Iteration 6	30	40

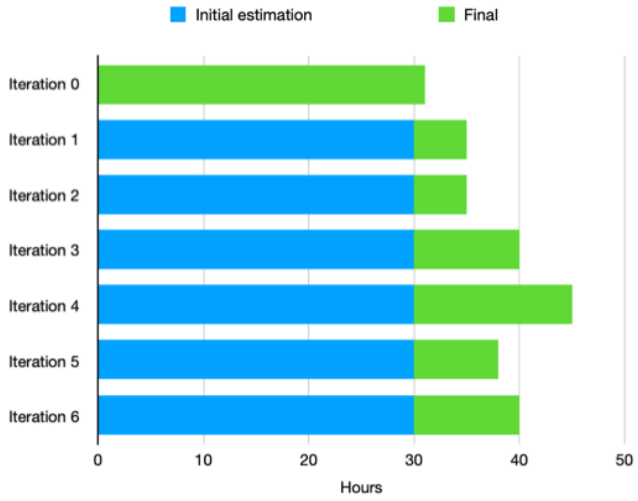


Figure 68 - Initial Vs. Final estimation of development time

Iteration 0 was special in the sense that we decided we would just take whatever time was needed to start the project with a solid foundation, time we took to develop the initial investigation

we can now find in the Chapter 2: State of the Art, decide which tools to use and methodologies we would follow as reflected in Chapter 4: Methodology, conducted a small poll to learn about our potential public, studied how we would structure this project and even developed a little proof of concept to make sure that what previously were just ideas could materialize into a real product.

In relation to the other iterations, the extra time we invested was mainly due to complications we did not initially foresee, extra functionalities which we felt like were needed at the time or simply just took more time than what we expected. This pattern repeated itself with this project memoir, which we developed in parallel to the iterations, and which writing was more intricate and required more time than what we initially considered, causing us to invest a considerable amount of extra hours to polish it and achieve the desired result, as seen in the following figure:

	Initial Estimation	Final
Memoir	120	156

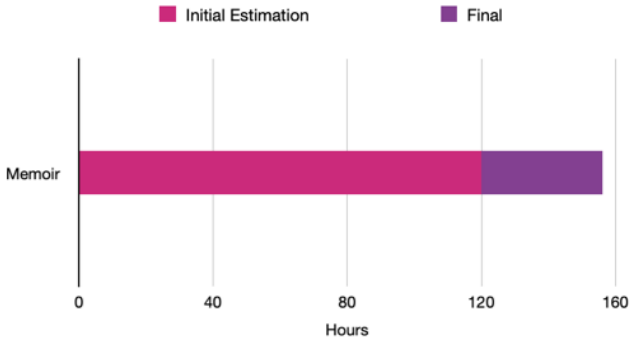


Figure 69 - Initial Vs. Final estimation time to write Memoir

With that said, we must also mention that after all the work, we have managed to meet all the objectives and requirements established for this project, obtaining a result the client is more than satisfied with. But, as we already know, we considered the client figure to be a combination of both the project director as well as the *Grupo Técnico de Coordinación (GTC)* of Gran Canaria, so we did not want to miss the opportunity of presenting them all the work we had done and ultimately find out if they could make some use of it.

In this way, we managed to set up a meeting with this group on the 24th of July of 2020, which to our surprise, not only assisted members of the GTC but also other people which were interested in the project/initiative we were taking. Between this people there were some with really important roles such as Nieves Martínez Cía, coordinator of GESCE, Cristóbal Nuez García, representative of the *Dirección General de Ordenación e Innovación Educativa del Gobierno de Canarias,* Hugo Fernández Ruíz, coordinator of *Programa de Familias y Participación Educativa de la Dirección General de Ordenación e Innovación Educativa del Gobierno de Canarias* or María Luisa Naranjo Báez, coordinator of Pediatrics and worker of the *Gerencia de Atención Primaria* (body that monitors, manages and controls the *Atención Primaria* services), just to name a few. You can find the full list of assistants and their roles in the Annex section 10.11 Attendees at the July 24, 2020 meeting.

For this presentation we prepared a series of explanatory videos which demonstrated how the Creation Platform worked and what results could be then obtained in the Visualizer, with the objective of making the process as simple as possible so that everyone understood what it was about and even more important, all the flexibility and possibilities it could entail.

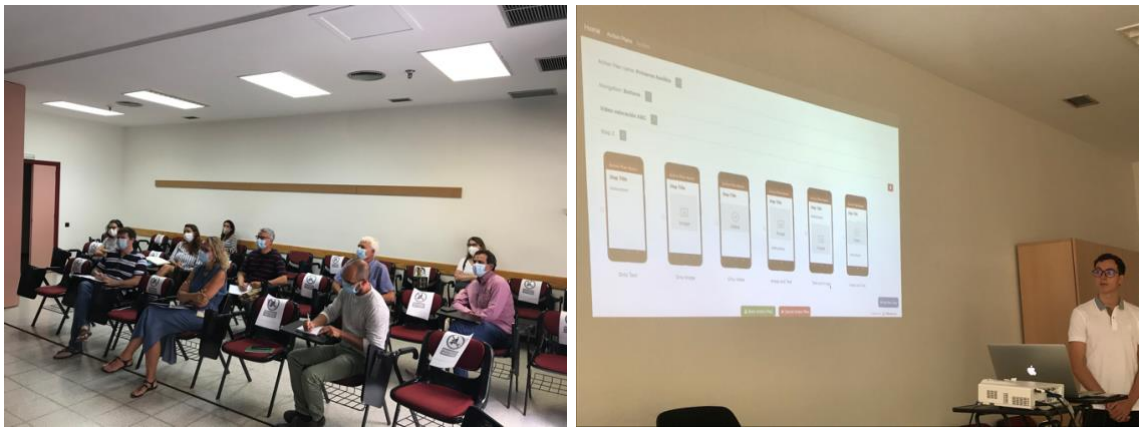


Figure 70 - Project presentation 24th of July of 2020

The presentation was, to my point of view, a great success because there were no doubts about what was explained, which meant that the product we had developed was intuitive and easy to understand. In addition, many commented on ways they could create content in relation to their areas of work, which means we have obtained that desired flexibility so that anyone can create the guides or plans they desire.

On top of this, we also received some feedback or further questions/considerations like if it will be available in iOS, the possibility of including audio as an option in the step types for people that cannot or do not want to read, and also expressed their concerns with possible legal aspects in relation to the *Ley Orgánica de Protección de Datos*, in the case of private Guides that could contain sensible or protected information, fact which we had not foreseen so it was a very nourishing experience for all of us.

As for the future of the project, they all agreed that it was a very interesting and innovative system to use and promote due to its adaptability and depth of options, making it a good starting point for the renovation of GESCE and for any possible future projects that might occur, establishing a relationship to continue working in the coming times.

8. Chapter 8: Conclusions

In this project we have left our comfort zone and worked in areas we had little knowledge of and faced problems we did not have the answers to, but thanks to our consistent work and the will to always keep going forward, we managed to create a product with a very bright projection for the future.

One of the best decisions taken during the development of this project, was right at the start of it, where we decided that developing a simple digital guide as an Android application was just not enough nor too appealing. Therefore, we went beyond our client's expectations and extended this simple idea into the development of a platform which anyone could use to create their own digital guides which could then be instantly portrayed in their mobile phones.

This model presents a clear advantage over the traditional approach of directly developing the mobile app, and the main reason why is, that we also provided the platform with the tools needed to edit, update and complete the content at any time, giving creators the ability and responsibility of managing their content as they wish. This has two main positive points: the first one is that during development we could completely focus on enhancing the components themselves and did not need to invest time searching and populating the application with the required information about emergencies, which was especially useful since we do not even have the correct knowledge to do so. The second positive point is that opposed to the more traditional route, we will not have to be contacted in the future if, for example, someone wants to delete a comma in one of their steps or wants to change the title to capital letters, relieving our job as developers and giving more power to the users.

Another advantage is that we have abstracted the problem in such a way that although it was designed for the representation of emergency guides, even our users have manifested that the system could be used for much more than that, being the possibilities literally endless: back to school guides, action plans specifically designed for someone with a rare disease, recipe books, interactive agendas with images, photo albums… the tools allow you to create any type of content which you then want to check in your mobile phone.

We have also managed to create a system whose components are properly isolated and do not depend on each other. For example, the Creation Platform works entirely on its own, it just needs a database to connect to, and the Visualizer could be connected to any API as long as they return the appropriate information in the expected format. The best part of this is that introducing a new component, such as the iOS version of the Visualizer for example, would be very simple and manageable to do: we would just need to build the application (probably trying to emulate its Android counterpart) and just, by calling the corresponding APIs to populate it, we would already have the applications to cover practically 99% of the mobile market.

In the process we have also tried to make all the components as scalable as possible, always leaving room for future enhancements since we are aware that this is just a prototype and can be improved in many ways. Some improvements we would like to implement in a plausible future version, which are documented in the remaining User Stories (7.2, 9, 12.1, 14, 16, 17) could be:

- Include new step types, specially one with the option of phone calls, since we think that it introduces many interesting possibilities.
- Adding the option of using *YouTube* videos, since it will save us lots of storage and would be a more suitable option for creators.
- Enhancing the functionalities of the Visualizer, such as the introduction of favorite Guides and Action Plans to for example always keep them on the top of the lists for easy access, the inclusion of search bars to facilitate finding the content you need, or a mode where custom colors can be deactivated in case the user cannot read the content correctly.
- It would also be very interesting to introduce the rating system, where specific authorized users could validate the procedures described in Action Plans as the correct/appropriate ones, providing a quality seal to that content. In addition, we could also integrate an option where the real users rate the content. On one side, this could help creators to decide for example, between two Action Plans which both cover the same emergency situation and on the other side, consumers can be certain that the information they are looking at is also backed up by other users.

We must also thank the GTC team for their continuous support and feedback. Their contribution in usability tests, questionnaires, and meetings was really beneficial for the development and final result of the project. Their outside view provided us with the much-needed information of how real users will behave when using the system, so we could specifically tailor it for them, making everything as convenient as possible for our users to work with.

On a final note, there is a great sense of satisfaction about completing a project of this magnitude, with which we have put to the test all the knowledge and experience acquired throughout our studies in the USJ and even learnt many new things from the process, but most importantly, we are really proud of creating a product whose main objective is the search for a more aware and caring society. Remember, saving lives is in everyone's hands.

9. Chapter 9: Bibliography

[1] N. Martínez Cía, Á. Cansino Campuzano, A. Cubas Medina, E. Martín Sánchez, S. González Campos and M. Artiles Suárez, "Guía De Emergencias Sanitarias En Los Centros Educativos," CONSEJERÍA DE EDUCACIÓN, UNIVERSIDADES, CULTURA Y DEPORTES Dirección General De Ordenación, Innovación y Promoción Educativa, 2 noviembre 2011. [Online]. Available: www.enfermeriacanaria.com/wptfe/wp-content/uploads/guia_emergencias_sanitarias.pdf. [Accessed June 2020].

[2] Cisco, "cisco.com," 9 Marzo 2020. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html. [Accessed June 2020].

[3] NAAXPOT SL, Naxxpot, May 2019. [Online]. Available: https://www.naaxpot.com/aplicaciones/urg-app/. [Accessed June 2020].

[4] Gobierno Vasco, "App 112 SOS Deiak," Gobierno Vasco, 13 May 2019. [Online]. Available: https://www.euskadi.eus/web01-a2bapps/es/contenidos/informacion/app_sosdeiak/es_tecnol/index.shtml. [Accessed June 2020].

[5] Global Disaster Preparedness Center, "Universal App Program," Global Disaster Preparedness Center, 26 January 2016. [Online]. Available: https://www.preparecenter.org/activity/universal-app-program/. [Accessed June 2020].

[6] Noten, "Manual de Primeros Auxilios - OFFLINE," Goolge Play, 24 March 2020. [Online]. Available: https://play.google.com/store/apps/details?id=ferdari.primeros_auxilios&hl=es. [Accessed June 2020].

[7] Ribera Salud, "App Store: YO Primeros Auxilios Ribera Salud," Ribera Salud, 29 February 2016. [Online]. Available: https://apps.apple.com/es/app/yo-primerosauxilios-ribera-salud/id1084969678. [Accessed June 2020].

[8] Statcounter, "Mobile Operating System Market Share Worldwide," StatCounter, May 2020. [Online]. Available: https://gs.statcounter.com/os-market-share/mobile/worldwide. [Accessed June 2020].

[9] Android, "Distribution Panel," Android, 1 June 2020. [Online]. Available: https://developer.android.com/about/dashboards. [Accessed June 2020].

[10] Apple, "App Store Developer Support," Apple, 27 January 2020. [Online]. Available: https://developer.apple.com/support/app-store/. [Accessed June 2020].

[11] W3Techs, "Usage statistics of client-side programming languages for websites," Web Technology Surveys, 06 June 2020. [Online]. Available: https://w3techs.com/technologies/overview/client_side_language. [Accessed June 2020].

[12] Bootstrap, "Build fast, responsive sites with Bootstrap," Bootstrap, June 2020. [Online]. Available: https://getbootstrap.com/. [Accessed June 2020].

[13] Zurb Foundation, "Foundation," Zurb, June 2020. [Online]. Available: https://get.foundation/. [Accessed June 2020].

[14] Semantic Org, "Semantic UI," Semantic Org, June 2020. [Online]. Available: https://semantic-ui.com/. [Accessed June 2020].

[15] Pure.css, "Pure.css," June 2020. [Online]. Available: https://purecss.io/. [Accessed June 2020].

[16] Ulkit, "Get Ulkit," Ulkit, June 2020. [Online]. Available: https://getuikit.com/. [Accessed June 2020].

[17] W3Techs, "Usage statistics of server-side programming languages for websites," Web Technology Surveys, 6 June 2020. [Online]. Available: https://w3techs.com/technologies/overview/programming_language. [Accessed 6 June 2020].

[18] Universidad de Salamanca, "¿A qué equivale el crédito ECTS?," Universidad de Salamanca, 18 November 2014. [Online]. Available: https://www.usal.es/que-equivale-el-credito-ects. [Accessed June 2020].

[19] M. Cohn, "Differences Between Scrum and Extreme Programming," Mountain Goat Software, 6 April 2007. [Online]. Available: https://www.mountaingoatsoftware.com/blog/differences-between-scrum-and-extreme-programming. [Accessed June 2020].

[20] Visual Paradigm, "Extreme Programming (XP) vs Scrum," 2020. [Online]. Available: https://www.visual-paradigm.com/scrum/extreme-programming-vs-scrum/. [Accessed 2020 June].

[21] Safety Seo, "Android App Development Company in Surat," Safety Seo, February 2020. [Online]. Available: https://safetyseo.com/android-app-development/. [Accessed June 2002].

[22] freepng, "Servidor Web, Los Servidores De Un Ordenador, Servicio De Web Hosting PNG," freepng, June 2020. [Online]. Available: https://www.freepng.es/png-reluwp/. [Accessed June 2020].

[23] M. Iqbal, "Pixabay," Pixabay, 18 May 2018. [Online]. Available: https://pixabay.com/es/illustrations/dise%C3%B1o-web-dise%C3%B1o-web-sitio-web-3411373/. [Accessed 2020 June].

[24] RRZEicons, "File:Database-mysql.svg," 3 June 2008. [Online]. Available: https://commons.wikimedia.org/wiki/File:Database-mysql.svg#filelinks. [Accessed June 2020].

[25] PNGOCEAN, "Logo apache http servidor apache software foundation servidor de servidores servidor web," [Online]. Available: https://www.pngocean.com/gratis-png-clipart-dohvn. [Accessed June 2020].

[26] Uneweb, "Consigue degradados para tu web con CSS3," Uneweb, 20 December 2013. [Online]. Available: http://tecnologiaenvivo.com/consigue-degradados-para-tus-web-con/. [Accessed August 2020].

[27] Silver sites, "Cómo cargar un archivo JS externo en HTML," Silver sites, 5 February 2019. [Online]. Available: https://www.silversites.es/cargar-js-externo-en-html/. [Accessed August 2020].

[28] C. Viebrock, "Official PHP Logo," 11 January 2001. [Online]. Available: https://es.wikipedia.org/wiki/PHP#/media/Archivo:PHP-logo.svg. [Accessed August 2020].

[29] Mel Hawthorne, "What are VOB Files?," Technipages, 13 April 2020. [Online]. Available: https://www.technipages.com/vob-files. [Accessed August 2020].

[30] Stack overflow, "Storing Images in DB - Yea or Nay?," June 2009. [Online]. Available: https://stackoverflow.com/questions/3748/storing-images-in-db-yea-or-nay. [Accessed June 2020].

[31] Stack overflow, "Saving images: files or blobs?," May 2010. [Online]. Available: https://stackoverflow.com/questions/1347461/saving-images-files-or-blobs. [Accessed June 2020].

[32] FrancescoBorzi, "FrancescoBorzi / Bootstrap-image-upload-form," 3 March 2015. [Online]. Available: https://github.com/FrancescoBorzi/Bootstrap-image-upload-form/blob/master/upload-image.php. [Accessed June 2020].

[33] ban-geoengineering, "Return a PHP page as an image," 7 November 2014. [Online]. Available: https://stackoverflow.com/a/26811487/11790814. [Accessed 26 April 2020].

[34] Bootstrap, "Collapse," 2020. [Online]. Available: https://getbootstrap.com/docs/4.0/components/collapse/. [Accessed June 2020].

[35] syanaputra, "Custom Login, Registration & Forgot Password," Bootsnip, 27 May 2015. [Online]. Available: https://bootsnipp.com/snippets/X04B0. [Accessed June 2020].

[36] S. Saurel, "How to retrieve an Unique ID to identify Android devices ?," Medium, 9 February 2017. [Online]. Available: https://medium.com/@ssaurel/how-to-retrieve-an-unique-id-to-identify-android-devices-6f99fd5369eb. [Accessed June 2020].

[37] S. Arciszewski, "PHP random string generator," Stack Overflow, 29 June 2015. [Online]. Available: https://stackoverflow.com/a/31107425/11790814. [Accessed June 2020].

[38] SudoPlz, "StackOverflow," 5 January 2017. [Online]. Available: https://stackoverflow.com/a/41491220/11790814. [Accessed June 2020].

[39] P. O., "What are the ways to programmatically generate Material Design color sets?," StackOverflow, 27 April 2016. [Online]. Available: https://stackoverflow.com/questions/28012185/what-are-the-ways-to-programmatically-generate-material-design-color-sets. [Accessed June 2020].

[40] A. Jasmin, "Detect whether there is an Internet connection available on Android [duplicate]," Stack Overflow, 24 August 2015. [Online]. Available: https://stackoverflow.com/a/4239019/11790814. [Accessed June 2020].

[41] M. Bointon, "Using Gmail with XOAUTH2," Github, 16 April 2020. [Online]. Available: https://github.com/PHPMailer/PHPMailer/wiki/Using-Gmail-with-XOAUTH2. [Accessed June 2020].

[42] LimiLabs, "OAuth 2.0 with Gmail over IMAP for installed applications," LimiLabs, February 2020. [Online]. Available: https://www.limilabs.com/blog/tag/xoauth2. [Accessed June 2020].

[43] Glassdoor, "Buscar sueldos y remuneración," Glassdoor, August 2020. [Online]. Available: https://www.glassdoor.es/Sueldos/index.htm. [Accessed August 2020].

[44] La Información, "¿Cuál es el coste real de tener un trabajador para una empresa?," La Información, 01 February 2019. [Online]. Available: https://www.lainformacion.com/practicopedia/cual-es-el-coste-real-de-un-trabajador-para-una-empresa/6491464/#:~:text=Se%20puede%20dividir%20en%2012,pagas%20en%20un%20mismo%20a%C3%B1o.&text=Las%20paga%20la%20empresa%20para,26.300%20euros%20a%20la%20empresa. [Accessed August 2020].

[45] Glassdoor, "Sueldos de Frontend Developer," Glassdoor, 21 August 2020. [Online]. Available: https://www.glassdoor.es/Sueldos/frontend-developer-sueldo-SRCH_KO0,18.htm. [Accessed August 2020].

[46] Glassdoor, "Sueldos de Backend Developer," Glassdoor, 23 August 2020. [Online]. Available: https://www.glassdoor.es/Sueldos/backend-developer-sueldo-SRCH_KO0,17.htm. [Accessed August 2020].

[47] Glassdoor, "Suelos de Android Developer Senior," Glassdoor, 25 August 2020. [Online]. Available: https://www.glassdoor.es/Sueldos/android-developer-senior-sueldo-SRCH_KO0,24.htm. [Accessed August 2020].

[48] Glassdoor, "Sueldos de Tester," Glassdoor, 23 August 2020. [Online]. Available: https://www.glassdoor.es/Sueldos/tester-sueldo-SRCH_KO0,6.htm. [Accessed August 2020].

[49] Glassdoor, "Sueldos De Project Manager," Glassdoor, 31 August 2020. [Online]. Available: https://www.glassdoor.es/Sueldos/project-manager-sueldo-SRCH_KO0,15.htm. [Accessed August 2020].

[50] Jazztel, "Fibra 100Mb + Fijo. Solo pagas por lo que hablas," Jazztel, August 2020. [Online]. Available: https://www.jazztel.com/tarifa/fibra-100mb-sin-llamadas?utm_source=labellium&utm_medium=sem&utm_campaign=generica&utm_term=pc&utm_content=internet&utoranw=labellium_sem_generica_pc_internet&gclid=CjwKCAjwkJj6BRA-EiwA0ZVPVuOpIvf8M0jXbLjSzaZ1CINvX-EdI4DQr8W. [Accessed August 2020].

[51] 000Webhost, "Upgrade plan," 28 August 2020. [Online]. Available: https://es.000webhost.com/members/upgrade. [Accessed 28 August 2020].

[52] MDB, "Carousel. Bootstrap Carousel," 2020. [Online]. Available: https://mdbootstrap.com/docs/jquery/javascript/carousel/. [Accessed June 2020].

[53] Bootstrap, "Carousel," 2020. [Online]. Available: https://getbootstrap.com/docs/4.0/components/carousel/. [Accessed June 2020].

10. Chapter 10: Annex

10.1. Final project proposal

Nombre alumno:	Bryan del Cristo Pérez Ramírez
Titulación:	Grado en Ingeniería Informática
Curso académico:	2019-2020

1. TÍTULO DEL PROYECTO

Guía de emergencias digital.

(Guía rápida de actuación ante emergencias sanitarias en los centros educativos)

2. DESCRIPCIÓN Y JUSTIFICACIÓN DEL TEMA A TRATAR

El proyecto consiste en el desarrollo de una aplicación para el sistema operativo Android, en la que se recogerán los conocimientos necesarios para la correcta actuación ante situaciones de emergencia, especialmente enfocado a los centros educativos. Como base, se utilizará la guía actualmente existente que fue desarrollada y promovida por el Gobierno de Canarias. Además, se contará con un servicio de backend para poder personalizar la guía con respecto a distintas instituciones.

3. OBJETIVOS DEL PROYECTO

- Estudio de las guías de emergencia convencionales.
- Adaptación de la guía existente al formato digital.
- Desarrollo de un backend que permita personalizar la aplicación.

4. METODOLOGÍA

La metodología se fijará en las primeras reuniones con el tutor.

5. PLANIFICACIÓN DE TAREAS

La metodología se fijará en las primeras reuniones con el tutor.

6. OBSERVACIONES ADICIONALES

El tutor del proyecto será Jaime Ignacio Font Burdeus.

10.2. Iteration 0

10.2.1. New User Stories

Back to above.

ID	1				
TITLE	**Create Action Plans for emergencies through web form (only text).**				
DESCRIPTION	Emergencies can be of any type: suffocation, heart attack, anxiety attack, etc. As a content creator, I want to define a series of steps that a person would need to follow to overcome the given emergency situation. This will be known as the Action Plan. At minimum, we want the action plan to be composed of just text entries, where the number of steps is variable. For example: Emergency Action Plan for Panic Attacks: • Step 1: Take in a slow, deep breath through your nose while counting to five. • Step 2: When you reach the count of five, let the breath out slowly (through your nose) at the same rate. • Step 3: Relax your muscles. Find a comfortable position to sit in (or lie down). • …				
PRIORITY	HIGH	**RISK**	LOW	**ESTIMATE**	4 h

ID	2				
TITLE	**Create Guides through web form (collection of Action Plans).**				
DESCRIPTION	In another web form, we want to create complete Guides. A Guide is composed of: a name and a number of already defined Action Plans. For this purpose, we want to see a full list of all Action Plans and be able to select the ones we want to be included in our guide. For example: • Suffocation: NO • Heart attack: YES • Anxiety attack: YES • Bleeding: NO • Epileptic seizure: YES				
PRIORITY	HIGH	**RISK**	LOW	**ESTIMATE**	4 h

ID	**3**				
TITLE	**Store Guides and Action Plans in persistent storage.**				
DESCRIPTION	Guides and Action Plans created through the web forms must be saved in a database for later use. Information should be organized in a logical way and allow easy extraction of data.				
PRIORITY	HIGH	**RISK**	LOW	**ESTIMATE**	7 h

ID	**4**				
TITLE	**Show Guides with their Action Plans in mobile application.**				
DESCRIPTION	As a consumer, I want to access the created Guides through a mobile phone app. When a Guide is clicked it will take us to another view where we can find the different Action Plans it contains and when one of these are clicked, we are taken once again, to another view where we can find the steps to follow.				
PRIORITY	HIGH	**RISK**	MEDIUM	**ESTIMATE**	15 h

ID	**5**				
TITLE	**Improve Action Plan creation to allow steps with images and videos.**				
DESCRIPTION	Defining an Action Plan with just text may be enough but sometimes we just understand procedures better with visual representations. For this reason, we want the web form to allow the use of images and videos, which may also be accompanied by some text to further emphasize the correct procedure. These must be correctly stored in the database and seen in the mobile phone.				
PRIORITY	MEDIUM	**RISK**	MEDIUM	**ESTIMATE**	27 h

ID	**6**				
TITLE	**Improve Guide creation to allow personalization.**				
DESCRIPTION	The Guide creation form is too simple and their visual representation in Android is too generic. For this reason, we want to allow some personalization to differentiate them in the app, for example, by choosing some branding image or logo and also a color to be used as accent throughout the Guide's contents.				
PRIORITY	LOW	**RISK**	MEDIUM	**ESTIMATE**	10 h

ID	7				
TITLE	**Improve Action Plan creation to allow other media types, customize their layout in Android and add extra options.**				
DESCRIPTION	We have already introduced images and videos to our Action Plans, but we also want to include GIFs. Furthermore, we want the user to be able to choose the layout in which they will be shown in Android. Examples could be: only text, first text and then an image or first a video and then text or even video, text, image, ... Whatever combination they may want. Of course, the app needs to adapt to these choices and be represented accordingly. As extra options, the most requested one is the option to include the possibility of calling an emergency number with a simple button click.				
PRIORITY	MEDIUM	**RISK**	HIGH	**ESTIMATE**	27 h

ID	8				
TITLE	**Login / Authentication system.**				
DESCRIPTION	To control who is creating content, we need a register/login system, which may be composed of a simple email and password. After register, this information must be authenticated by at least email verification. Once logged in, this user will be able to create Action Plans and Guides, access its already created content, modify or even delete it if necessary.				
PRIORITY	MEDIUM	**RISK**	LOW	**ESTIMATE**	7 h

ID	9				
TITLE	**Feedback / Verification system**				
DESCRIPTION	Anyone with a valid email account may create Guides and Action Plans in our system. To provide some veracity to this content, we would like to have a system through which Action Plans can be verified as the correct procedure. With this in consideration, we would like that some users, for example, specialists in the subject such as nurses or doctors for example, have special access to this verification system in which they can go through all the different Action Plans and mark them as verified. If a plan is verified, we want some sort of mark to be shown when creating a guide, this should also be reflected in the database. In case the action plan is not correct, we would also like that experts can edit the information and correct it. These changes must be consistent throughout the project (web, DB and mobile). On the other hand, we would also like that consumers have the ability to give Action Plans and Guides a score, in a scale of 1 to 3 stars.				
PRIORITY	LOW	**RISK**	HIGH	**ESTIMATE**	30 h

10.3. Iteration 1

10.3.1. Tasks and Acceptance Tests

Back to above.

Tasks

US1

- Create a form that includes:
  - A name for the Action Plan.
  - Variable number of text inputs which will correspond to the steps.
  - Help texts to orient the user through the process.
  - Submit button.
- Use bootstrap to provide some styling to the website.

US2

- Create a form that includes:
  - A name for the Guide.
  - A list with all Action Plans already created.
  - A way to select from this list the plans to include in the Guide.
  - Help texts to orient the user through the process.
  - Submit button.
- Use bootstrap to provide some styling to the website.

US3

- Design the different data models and their relationships.
- Create the database.
- Create the tables corresponding to the design.
- Create the required PHP scripts to upload Action Plans and Guides to the database.
- Avoid incomplete or malicious upload attempts.

US4

- Create the APIs needed in PHP to retrieve the information from the database and return it in JSON format.
- Study and find the best structure of this JSON so that it can be interpreted correctly in Android.

In Android Studio:

- Create the data models to convert the JSON into objects we are able to work with.
- Create the script to connect to the internet and call the APIs to retrieve the information.
- Create the activities for Guides, Plans and Steps. Each of them should have:

- A Layout that includes a ListView.
- An adapter to populate each ListView with the corresponding information.
- The logic to move from one activity to the next. The order is Guides -> Plans -> Steps. Clicking on a Step does nothing.

Acceptance tests

As a creator I want to:

- Be able to create an Action Plan named "Plan 1" composed of two steps. One needs to say: "This is an example text" and the other one must include numbers and special characters such as '*\#@/.
- This plan and steps must be stored correctly in the database.
- Be able to decide the number of steps an Action Plan should have.
- While creating Action Plans, choose to have one step and write "1" as its text description. Then change number of steps to two. "1" should still be written in Step 1.
- While creating Action Plans, if we leave any of the sections blank, we should not be able to upload it to the database.
- Be able to create a guide called "Guide 1" which includes the Plan 1 we created before.
- This guide must be stored in the database.
- While creating Guides, all the already created Action Plans may be chosen.
- While creating Guides, if no name is provided or there is no plan selected, guide should not be uploaded.

As a consumer/creator I want to:

- Be able to see all the created guides in my mobile phone.
- If we click on Guide 1, we should see the Action Plans it contains, so Plan 1 must appear.
- If we then click on Plan 1, we should see the two steps we mentioned in the first test.
- If we close the app and new guides are created, we must seem them when launching the app again.
- If a guide is deleted from the database, it should not be shown when the application is restarted.
- If a plan is deleted from the database, it should not be present inside the guides which included it when the application is restarted.

10.3.2. New User Stories

Back to above.

ID	**10**				
TITLE	**General view for guides and action plans to offer at least CRUD functionality.**				
DESCRIPTION	In order to manage all the content created, we want to create new views for Guides and Action Plans where at minimum, we will be able to create, read, update and delete them (also known us CRUD). Any further options we come up with in the future can be implemented in these general views such as creating filters to find the searched for content. The different options should be easily distinguishable by the user and should be as error proof as possible, giving the user the opportunity to rectify in case of misunderstanding or mistake. For example, there should be a double confirmation before deleting any content.				
PRIORITY	MEDIUM	**RISK**	HIGH	**ESTIMATE**	15 h

ID	**11**				
TITLE	**Home screen.**				
DESCRIPTION	We want a simple home screen that will act as landing page. It should include a brief description of what the Creation Platform is and include some navigation option to move to the general views for Guides and Action Plans.				
PRIORITY	MEDIUM	**RISK**	LOW	**ESTIMATE**	3 h

ID	**12**				
TITLE	**Caching in Android app.**				
DESCRIPTION	Once the application works with all the needed data, we should investigate the different caching options available for our system. Android offers some solutions such as the use of internal files, Shared Preferences or even a small database known as SQLite. These are just some options, we should search for the one that suits as the most and implement it. The main objective is to avoid the download of the same content each time we start the app and more importantly, be able to access the content without internet connection.				
PRIORITY	MEDIUM	**RISK**	HIGH	**ESTIMATE**	15 h

10.4. Iteration 2

10.4.1. Tasks and Acceptance Tests

Back to above.

Tasks

US5

- Modify database tables to be able to create new types of steps with images and videos.
- Modify create Action Plan form to:
  - Include a way to create steps composed of only images and only videos.
  - Create the necessary inputs for them and include frontend validation.
- Modify POST functionality to upload media.
  - Valid images must not exceed 5MB and be of type *png, jpg* or *jpeg*.
  - Valid videos must not exceed 15MB and be of type *mp4.*
- Create the necessary APIs to retrieve media.
- Investigate how to show media in Android and implement it.

US11

- The home screen should include:
  - A title.
  - A description of the functionality of the platform.
  - Buttons that link to our previously created scripts to create Guides and Action Plans.
- Update navigation bars throughout platform so that we are taken to this screen when clicking on "Digital Guide Creator".

Acceptance tests

- Create an Action Plan composed of two valid images. Check the images and Action Plan have been saved correctly.
- While creating an Action Plan, select to have two images but leave the inputs empty. When Create button clicked, plan should not be uploaded, plus user should be alerted of it.
- Create an Action Plan with an invalid image. When Create button clicked a different alert should pop up to notify the user.
- Create an Action Plan with two images, one valid and the other one invalid. Action Plan and images should not be saved. Not valid alert should pop up.

- Create a Guide with the valid plans created and show them in Android. Images must be viewed correctly.

Do the same process but with videos.

- Home page must include the elements described in the task.
- In the home page check that the buttons take you were supposed to if clicked.
- In the Action Plan creation page, check that the navigation bar element takes you to home page if clicked.
- In the Guide creation page, check that the navigation bar element takes you to home page if clicked.

10.4.2. New User Stories

Back to above.

ID	**5.1**				
TITLE	**Fix Action Plan creation so that all images/videos have to be valid to upload.**				
DESCRIPTION	Solve error where if valid and invalid images are inserted in the Action Plan form, valid images are still uploaded when they should not. Fix the logic to check that all media has to be valid to upload the plan successfully.				
PRIORITY	HIGH	**RISK**	LOW	**ESTIMATE**	30 min

ID	**13**				
TITLE	**Buttons view for steps.**				
DESCRIPTION	Right now in Android, we just show steps in a simple ListView. We should investigate other ways this could be done but the client wants a specific one to be developed. A view where only one step will be showed per screen and to change the current step displayed, we will use two directional buttons: back and next.				
PRIORITY	MEDIUM	**RISK**	LOW	**ESTIMATE**	3 h

ID	**14**				
TITLE	**Improve options to upload videos in Action Plan creation.**				
DESCRIPTION	The only option right now to include a video in your steps is to upload it from your computer. We should study if content creators like this option or they would rather just provide the link to a YouTube video. In any case, we should include this option, especially for its advantages: easier for the creator and less memory consumption in our database. We should also implement in the Visualizer the use of *ExoPlayer*, a much better video player than the default one provided by Android, especially for its ability to stream YouTube videos.				
PRIORITY	LOW	**RISK**	HIGH	**ESTIMATE**	20 h

10.5. Iteration 3

10.5.1. Tasks and Acceptance Tests

Back to above.

Tasks

US5.1

- Fix the POST script to check that all images are valid before proceeding with upload to database.

US7

- Allow GIFs in Action Plan creation.
- Investigate how we can support the creation of custom steps both in Creation Platform and Visualizer.
- Implement the best solution.
- Allow phone call input in Action Plan creation.

US13

- Implement a new Activity that shows each step at a time and that can be changed with the use of two directional buttons.

Acceptance tests

- Create an Action Plan composed of two valid images. Check the images and Action Plan have been saved correctly.
- Create an Action Plan with two images, one valid and the other one invalid. Action Plan and images should not be saved. Not valid alert should pop up.
- Create an Action Plan with a GIF. Check it is properly saved and that we are able to see it correctly in the Visualizer.
- Creating an Action Plan, we are able to decide which inputs we want and in what order.
- If we leave any of the inputs empty, frontend validation should not allow us to submit the form.
- Try different customization options and verify they are uploaded correctly.
- These should be correctly represented in Android.
- Create an Action Plan with the option of calling a number, verify that if clicked the call works correctly.
- Create an Action Plan with multiple steps and with the Buttons option. Check that each of the steps is viewed in a separate screen as expected.
- Verify that buttons always work so try extreme cases as one, two and ten steps.

10.5.2. New User Stories

Back to above.

ID	**7.1**				
TITLE	**Fix Action Plan creation to improve UX, update backend and Visualizer.**				
DESCRIPTION	Client wants some fixes to be done in the Action Plan creation such as: modify the visibility of the cards so that only one appears at first and the following ones are shown as you fill the form, create a GIF to choose step navigation, change add new step, save/cancel plan functionality/placement, summarize the input inserted in each of the cards when closed, include the option to add a title to the steps and fix the navigation bar to the top. Apart from this, we have to adapt the backend of the website to upload these new types of plans and of course make the necessary changes to the Visualizer so that they are viewed properly.				
PRIORITY	MEDIUM	**RISK**	HIGH	**ESTIMATE**	23 h

ID	**7.2**				
TITLE	**Create new step type to include phone calls.**				
DESCRIPTION	Now that we have the ability to include new options to the step types selection, we want to include a new one where we will be able to provide a telephone number and a text to accompany it. The phone number must be callable in the Visualizer.				
PRIORITY	LOW	**RISK**	LOW	**ESTIMATE**	10 h

10.6. Iteration 4

10.6.1. Tasks and Acceptance Tests

Back to above.

Tasks

US7.1

- Use CSS's visibility property and the Accordion's open/close functionality to progressively show each section of the form.
- Create GIFs for navigation and show them in the Navigation card.
- Add new step button should be placed outside the accordion and when clicked it has to append a new step card after the last one.
- Remove step button should be placed inside each card but somewhere where the user will not click it accidentally. When clicked it should remove the corresponding card. We must change the previous functionality where only last step can be deleted. All of them must be removable except for one, to avoid creators being able to upload a plan without steps.
- Save Action Plan button should be placed at the bottom, outside the accordion, and when clicked must trigger the frontend validation to check all inputs have been filled. When valid, the submitting script must be called.
- Cancel Action Plan button should be placed next to Save button, each with different colors to convey they have different functionalities, and when clicked must reset the current Action Plan.
- Find the way to tell the user what they inserted in each of the cards when they are closed.
- Include step title for each of the step type possibilities.
- Fix navigation bar to the top so that it is always visible, even when we scroll down.
- Update database to adapt to all of these changes and hold the correct information.
- Update POST script so we are able to upload this new version of Action Plans.
- Investigate how to update the Visualizer to correctly view this new version of Action Plans.

US8

In the Creation Platform:

- Create the following sites:
    - Register
    - Login
    - Forget Password

- Registering requires: full name, alias, email, password and confirm password.
- When registering, the provided email must be verified.
- When registering, password must be at least five characters long.
- When registering, there cannot be two users with the same email nor alias.
- Login requires: verified email and corresponding password.
- Forget password requires: already registered email.
- In forget password, we have to provide a safe method for the user to change its password.
- In all the sites, capture possible error situations.

Acceptance tests

- At first when loading the Action Plan creation form, only the first card to insert the plan title will be visible. When written and OK button clicked, navigation card must appear, and the previous card should be closed. The header of the first card must be updated with the title inserted by the user. Check that the following cards follow the same pattern.
- Check that navigation GIFs are in a continuous loop. Open and close its card to verify it does not stop.
- When clicked, Add New Step button should append a card after the last one and it must show the step type options. Previous card should be closed.
- Verify that Remove Step button, removes the corresponding step and not others. With five steps try deleting the first, middle and last step. Steps left should be 2 and 4.
- When clicked, Cancel button must reset the current created plan.
- When clicked, Create button must verify that no input is left empty. Check leaving the input of each card empty, to verify this is done correctly. When all is filled, plan must be uploaded.
- Add six steps and select each of the step types and verify the title input is available in all the cases. Scroll the web page up and down to make sure the top navigation bar is always visible and on top of the rest of the content.
- Create a plan with Scroll navigation and six steps, one with each of the step types. Check in the Visualizer the result is the expected one.
- Change the previous plan to Button navigation (through database) and verify again that all elements are showed correctly.
- When registering, check that if any input is left empty, data is not uploaded.
- If email is already registered, user should be alerted.
- If alias is already registered, user should be alerted.
- Verify that passwords must be the same. Do not let the user continue till they are.
- Check that verification email is received correctly.

- When login, email and password may be correct, but until email is not verified do not let the user continue. Alert the user.
- Try common accounts as admin admin, and verify we do not log in.
- Try valid email and password and should be redirected to home screen.
- In forget password, verify that providing a valid email, you receive correctly the link to update your password.
- Change the password and verify in login that previous password does not work anymore but new one does.

10.6.2. New User Stories

Back to above.

ID	**7.3**				
TITLE	**Fix buttons of Action Plan creation form.**				
DESCRIPTION	The client wants to modify the position and text of the buttons of the Action Plan creation form. We should have three buttons, Save Action Plan, Cancel Action Plan and in other location away from these two, Add New Step. We should also add icons to the buttons to convey what clicking the button will do, which will improve the UX.				
PRIORITY	MEDIUM	**RISK**	LOW	**ESTIMATE**	30 min

ID	**15**				
TITLE	**Permission system.**				
DESCRIPTION	Now that we have creators in the platform, we should have a permission system to allow public or private content. We should study how this will affect the rest of our architecture, especially the Visualizer, since we will have to access the private content in a secure way.				
PRIORITY	MEDIUM	**RISK**	HIGH	**ESTIMATE**	15 h

10.7. Iteration 5

10.7.1. Tasks and Acceptance Tests

Back to above.

Tasks

US7.3

- Change buttons layout and the text inside them.
- Add icons to buttons to improve UX.

US10

- Create new view for Action Plans that is able to:
  - Show all the already created plans from the logged user.
  - For each of them allow their edition, duplication and deletion.
  - Include separate button to create new plan.
  - Show all the already created plans from other users but which have agreed to share it.
  - Study what we can allow other users to do with these plans.
- Do the same for Guides.

US15

- Study how we can create a permissions system and apply it to the created content. Update the Creation Platform to support this new possibility.
- Study how this affects the rest of the architecture and make the necessary changes to carry it out.

Acceptance tests

- Check that buttons have the new text and icons are displayed correctly.
- Create a new user and log in. Go to the Action Plans main view, no plan should be displayed, and no errors should occur.
- Create a new plan with three steps and save it. Edit it, and change its name to "Plan Edit", its navigation type to the one that is not selected and change all its steps contents but not the type. Check that the changes were done correctly.
- Edit again the same plan but now choose different step types for each of the steps. Check that it is uploaded correctly.
- Edit it again and remove all steps except one. Verify these steps do not exist anymore in the database and nor their files in the filesystem.
- Create a new plan that contains steps with files. Duplicate it. The new copy must appear in the table and should now appear as yours, this means you can edit, duplicate again and delete it.

- Edit this duplicate and change the files, check that they have been changed correctly in the database and filesystem.
- Duplicate the copy again and then delete it. It should disappear from the table.
- Go to the Guides main view and no Guide should appear. No error should occur either.
- Create a new Guide, select one plan and save it. Edit this Guide, change the name to My guide duplicate and select two plans now. Check that it has been updated correctly.
- Edit it again and uncheck the selected plans. Frontend validation should not allow us to submit it.
- Delete this Guide. It should disappear from the table.

The following tests were written after we decided how were going to proceed with the permissions system:

- Create a public and a private Action Plan. Create a new user and log in with it. In the Action Plans table, only the public plan should appear and as actions, only duplication should be possible.
- Duplicate it and check that you now have a modifiable copy you can work with.
- Create a public and a private Guide. In the Visualizer check that only the public Guide is shown in the public tab, can be selected and added. Get the code from the private Guide and insert it in the Visualizer. The corresponding Guide should be found and can be added to our list.
- Use the same code again. An alert message should appear telling us that it has not been found or it is already included.
- Check that codes smaller or greater than six characters in length are not sent to the API.
- Check that an invented, valid in length code, also displays the alert mentioned earlier.

10.7.2. New User Stories

Back to above.

ID	**10.1**				
TITLE	**Edit Guides.**				
DESCRIPTION	We just want to replicate the same functionality of editing done in Action Plans but adequate to the new create guides form we will develop.				
PRIORITY	MEDIUM	**RISK**	LOW	**ESTIMATE**	1 h

ID	**16**				
TITLE	**Improved files management.**				
DESCRIPTION	Duplicating files in our filesystem is not a good idea at all. We want to make the necessary changes to make it more efficient and also improve it: create a file selector where creators can view all the files they have uploaded and will have the ability to choose from one of these or just upload new ones. In this way the content can be reused, and it is much more efficient for our system.				
PRIORITY	LOW	**RISK**	HIGH	**ESTIMATE**	25 h

ID	**17**				
TITLE	**Guides and Action Plans tables size control.**				
DESCRIPTION	As these tables will grow in size very quickly, we should investigate further ways we could limit their size, like include pagination so that only a set of rows is showed at a time, or more advanced filtering like: only show our content, only other's content, filter by number of plans it includes (in the case of guides), etc.				
PRIORITY	LOW	**RISK**	LOW	**ESTIMATE**	10 h

ID	**18**				
TITLE	**Move Creation Platform to real server.**				
DESCRIPTION	We would like to make the platform available to real users and to do this, we should move the Creation Platform from our local machine into a real server, with all the necessary changes to make it work in the exact same way.				
PRIORITY	MEDIUM	**RISK**	LOW	**ESTIMATE**	4 h

10.8. Iteration 6

10.8.1. Tasks and Acceptance Tests

Back to above.

Tasks

US6

- Investigate the different possibilities we can offer to customize Guides in the Visualizer like adding a branding image or a custom color.
- Modify the Guide creation script to include these possibilities.
- Modify the Visualizer to adapt to these new changes.

US10.1

- When the Guide creation form is finished, we will be able to develop Guide editing, which needs to work in the same way as with Action Plans did.

US12

- Investigate how we can integrate a caching system in the Visualizer where the contents must only be downloaded once and only requested again when they have been updated.
- Find the best way to store the content in the mobile device, which must be easy to access and modify.
- Make the Visualizer work without internet connection.

US18

- Find a free hosting solution adequate to our project.
- Migrate the database to keep all our tables, relations and content.
- Verify that everything works as it did in our local machine.

Acceptance tests

- Create a Guide with a branding image and a custom color, check they are applied correctly in the Visualizer.
- Create a Guide with a branding image but no custom color, check they are applied correctly in the Visualizer. Default black color should be used.
- Create a Guide with no branding image but yes custom color, check there are no inconsistencies in the Visualizer.
- Create a Guide with neither branding image nor custom color, check there are no inconsistencies in the Visualizer.
- Edit the last Guide changing its title, number of plans, and now add a branding image and choose a custom color. Check that these changes have been applied correctly throughout our system.

- Create simple Action Plans and Guides and load the Visualizer once. Check that the content has been saved correctly in the device.
- Modify those Action Plans and Guides and load the Visualizer once more. Log the response to check that only the modified content has been downloaded again.
- Close the Visualizer and turn Airplane mode on (no internet). Open the Visualizer again and check that all the content we had already visualized can be accessed and works as expected, including all media types.

10.8.2. New User Stories

Back to above.

ID	**12.1**				
TITLE	**Advanced caching in Android app.**				
DESCRIPTION	We already have developed a base version of caching in the Android app, but this can be improved at minimum in these ways: the first is based on developing a more intelligent system where we do not download the entire Guide or Action Plan when it is modified, but only the exact parts that have changed. Secondly, it should not be as rigid as having internet or not; we could implement a system that checks the quality of your connection and if it is not good at the moment, we could wait for another time where it is better or just ask the consumer whether they want to download them now or later. This system could also be improved by finding the best way to also store videos, so we are able to access them independently of having a good/stable internet connection.				
PRIORITY	LOW	**RISK**	HIGH	**ESTIMATE**	25 h

10.9. Meeting Notes

MEETING:	**1**	
Date:	19/12/19	
Start time:	16:30	**End time:** 18:00
Location:	Universidad San Jorge	
Written by:	Bryan Pérez	
Attendees:	Jaime Font and Bryan Pérez	
Notes:	Client is satisfied with the results obtained: we have defined the requirements we want to achieve in this project, the main architecture of the system and even developed a small PoC to make sure we are on the right track and can keep moving forward. Talked about the next steps to follow. For the first iteration we will work on User Stories 1, 2, 3 and 4.	

MEETING:	**2**	
Date:	09/04/20	
Start time:	12:00	**End time:** 13:15
Location:	Online. Microsoft Teams.	
Written by:	Bryan Pérez	
Attendees:	Jaime Font and Bryan Pérez	
Notes:	Reviewed everything done in Iteration 1 and run acceptance tests. Talked about possible improvements that could be done to the Creation Platform and Android application. Wrote new User Stories. For the next iteration we will work on User Stories 5 and 11.	

MEETING:	**3**	
Date:	30/04/20	
Start time:	12:00	**End time:** 12:50
Location:	Online. Microsoft Teams.	
Written by:	Bryan Pérez	
Attendees:	Jaime Font and Bryan Pérez	

Notes:	Reviewed everything done in Iteration 2 and run acceptance tests. Talked about how the client would like to see new navigation options in the Android application and how we could improve the system to allow the use of YouTube videos throughout. Wrote new User Stories. For the next iteration we will work on User Stories 5.1, 7 and 13.

MEETING:	**4**	
Date:	14/05/20	
Start time:	12:00	**End time:** 13:00
Location:	Online. Microsoft Teams.	
Written by:	Bryan Pérez	
Attendees:	Jaime Font and Bryan Pérez	
Notes:	Half-Iteration 3 meeting to decide which direction we should follow: We should simplify the problem and just offer a closed set of step types. Web page can grow too much vertically so we must take that into consideration. Reduce the amount of explanatory text in creation forms.	

MEETING:	**5**	
Date:	21/05/20	
Start time:	12:00	**End time:** 13:20
Location:	Online. Microsoft Teams.	
Written by:	Bryan Pérez	
Attendees:	Jaime Font and Bryan Pérez	
Notes:	Reviewed everything done in Iteration 3 and run acceptance tests. Talked about how we could make filling the creation forms a little bit more dynamic and intuitive and other general improvements we could make. Wrote new User Stories. For the next iteration we will work on User Stories 7.1 and 8.	

MEETING:	**Test session with Ana Ramírez**
Date:	24/05/20

Start time:	17:00	**End time:** 18:50
Location:	Online. Skype.	
Written by:	Bryan Pérez	
Attendees:	Ana Ramírez and Bryan Pérez	
Notes:	Tested the Creation Platform to see if users understand how it works and find potential problems we had not thought of.	

MEETING:	**6**	
Date:	18/06/20	
Start time:	12:00	**End time:** 13:20
Location:	Online. Microsoft Teams.	
Written by:	Bryan Pérez	
Attendees:	Jaime Font and Bryan Pérez	
Notes:	Reviewed everything done in Iteration 4 and run acceptance tests. Small changes to the button placement of the creation forms and add icons to them. Talked about how could proceed with the permission system and how to structure the main views for Guides and Action Plans. Wrote new User Stories. For the next iteration we will work on User Stories 7.3, 10 and 15. Reviewed the memoir.	

MEETING:	**7**	
Date:	06/07/20	
Start time:	12:00	**End time:** 13:40
Location:	Online. Microsoft Teams.	
Written by:	Bryan Pérez	
Attendees:	Jaime Font and Bryan Pérez	

Notes:	Reviewed everything done in Iteration 5 and run acceptance tests. Talked about how editing guides is an action we must develop, how the filesystem should be improved, how the tables of the main views can grow too much and about the option of uploading the Creation Platform to a real server. Wrote new User Stories. For the next iteration we will work on User Stories 6, 10.1, 12 and 18. Reviewed the memoir.

MEETING:	**Project presentation to GTC and others**	
Date:	24/07/20	
Start time:	13:15	**End time:** 14:00
Location:	Edificio de Gerencia de Atención Primaria, Las Palmas de Gran Canaria.	
Written by:	Bryan Pérez	
Attendees:	Full list is in Annex section 10.11	
Notes:	Presented the project to a group of people with the idea of making it part of the process of transforming GESCE into the mobile format. We also got some feedback and general opinions that we could use to improve the overall result.	

MEETING:	**8**	
Date:	04/08/20	
Start time:	12:00	**End time:** 13:00
Location:	Online. Microsoft Teams	
Written by:	Bryan Pérez	
Attendees:	Jaime Font and Bryan Pérez	
Notes:	Reviewed everything done in Iteration 6 and run acceptance tests. Client is happy with the results obtained. The caching system could be further improved in a future version of the project. Wrote new User Story. Reviewed the memoir.	

10.10. Google Form. PFG Bryan Pérez: Guía digital de emergencias

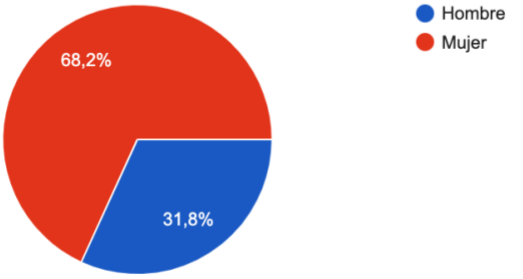
Back to above.

PFG Bryan Pérez: Guía digital de emergencias

201 respuestas

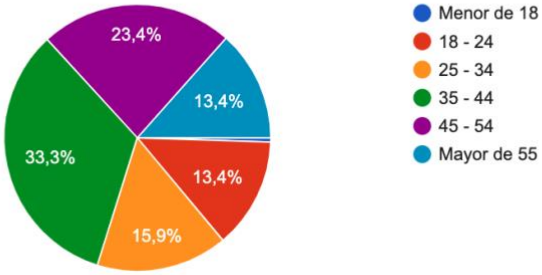
¿Eres..?

201 respuestas



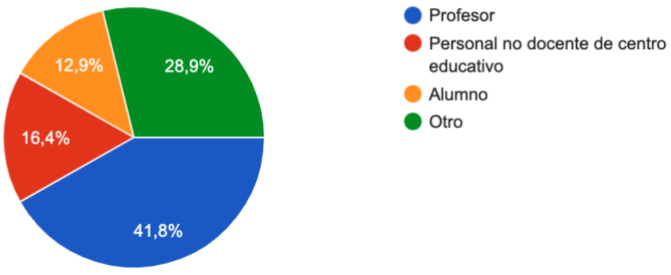
¿Podrías indicar tu edad?

201 respuestas



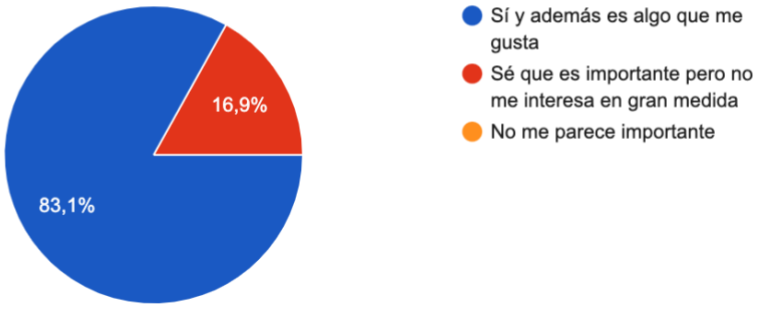
¿Eres..?

201 respuestas



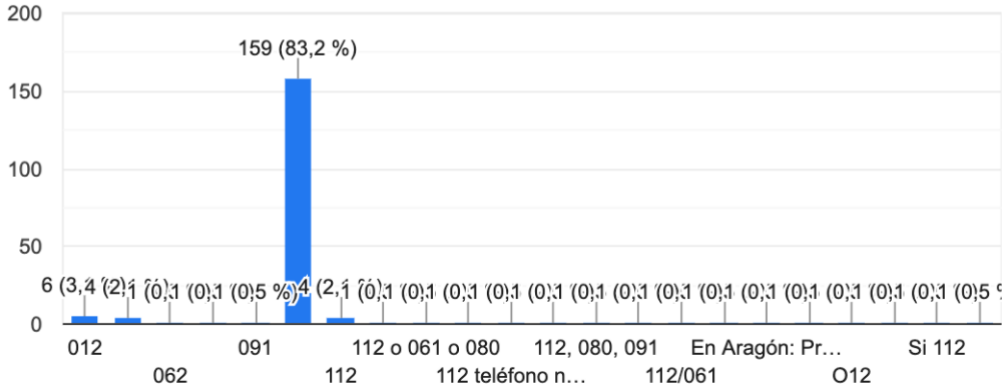
¿Te parece importante estar informado sobre cómo actuar ante emergencias?

201 respuestas



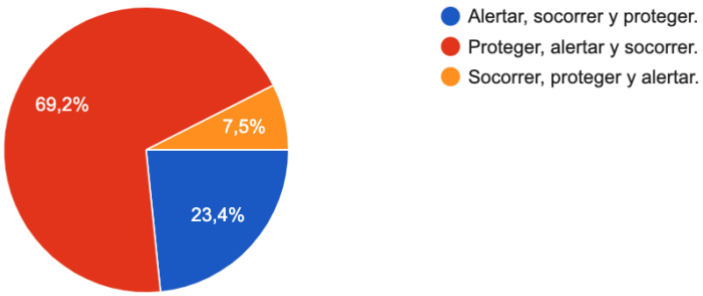
¿Sabrías a qué número llamar en caso de emergencia? En caso afirmativo, escríbelo a continuación.

191 respuestas



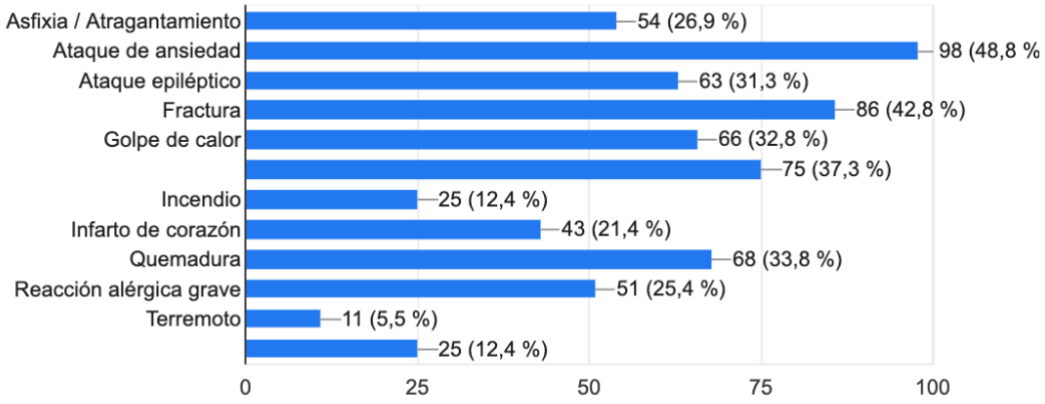
En caso de emergencia sanitaria, ¿sabrías en qué orden actuar?

201 respuestas



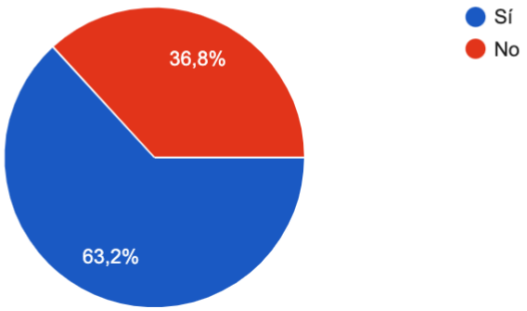
¿Has vivido o presenciado alguna de las siguientes situaciones?

201 respuestas



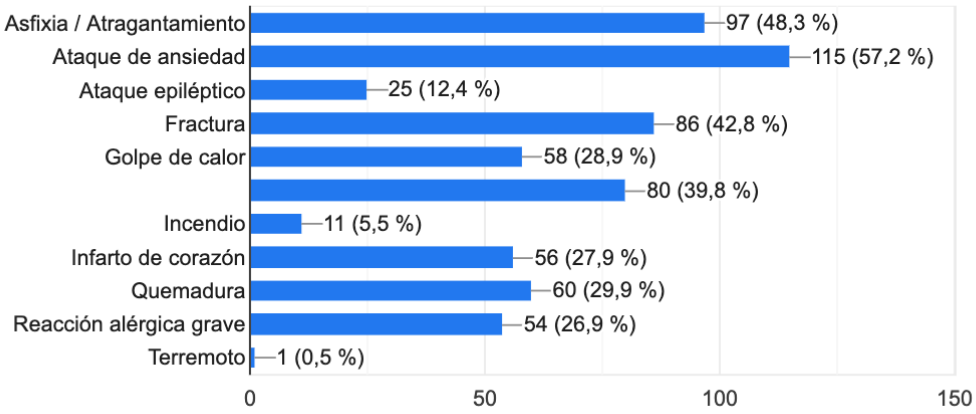
¿Has asistido a algún tipo de curso en el que se tratasen éstos u otros temas sobre actuación ante emergencias?

201 respuestas



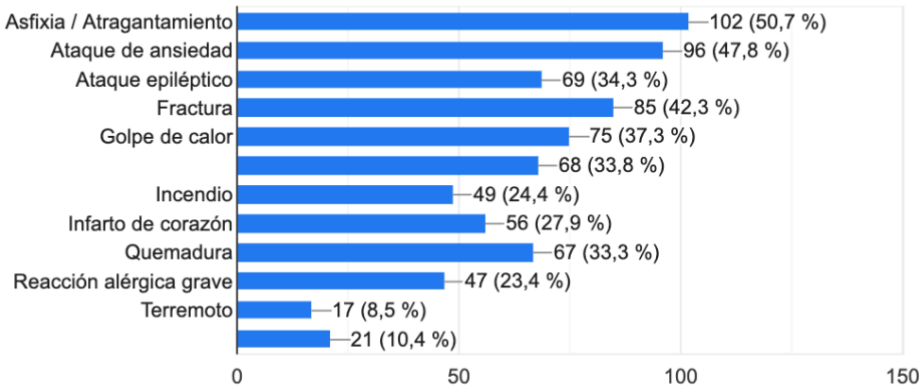
De estas situaciones, ¿cuáles consideras que ocurren más a menudo?

201 respuestas



¿Y ante cuáles crees que sabrías reaccionar adecuadamente?

201 respuestas



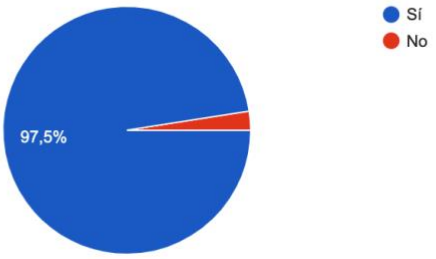
Aparte de los temas ya mencionados, ¿Existe algún otro en el que tengas especial interés o te gustaría conocer en mayor profundidad?

65 respuestas

- La diabetes y los ataques epilépticos
- Manejo de personas con discapacidad
- Todos
- asistencia a bebés
- Torcerse un tobillo
- Reanimación cardiopulmonar
- Inundación; Temporal de viento; Contaminación por radiación
- Otros tipos de emergencia que se puedan producir en la USJ: Fuga de gases o sustancias nocivas (laboratorios,...), tumulto, ataque terrorista...
- RCP

¿Crees que sería útil una aplicación móvil que indicase qué hacer en estos y otros muchos casos?

201 respuestas



10.11. Attendees at the July 24, 2020 meeting: *Coordinación y presentación de la Plataforma para la digitalización de la Guía de Emergencias sanitarias.*

- Nieves Martínez Cía: *Coordinadora de la Guía de emergencias sanitarias. Enfermera del Centro de Salud de Las Remudas, Telde. Participante del Proceso Comunitario de Las Remudas.*
- Ana Isabel Santana Arrocha: *Directora del Centro de Educación del Profesorado de Telde.*
- Ángeles Cansino Campuzano: *Pediatra y coordinadora de Pediatría de Atención Primaria.*
- María del Mar Artiles Suárez: *Enfermera.*
- Leticia Curbelo Delgado: *Coordinadora de proyectos de la fundación ADSIS Canarias, participante del Proceso Comunitario de Las Remudas.*
- Cristóbal Nuez García: *Representante de la Dirección General de Ordenación e Innovación Educativa del Gobierno de Canarias. Coordinador de los Centros de Educación del Profesorado de Canarias.*
- Antonio Cubas Medina: *Enfermero.*
- Artemi Dámaso Manzanares: *Enfermero y matrón.*
- Santiago González Campos: *Enfermero y docente de la Universidad de Las Palmas.*
- María Luisa Naranjo Báez: *Pediatra de Atención Primaria y Coordinadora de Pediatría. Trabajadora de la Gerencia de Atención Primaria.*
- Enrique Martín Sánchez: *Médico coordinador del Servicio de Urgencias Canario (SUC) de Las Palmas.*
- Ana María Ramírez Reina: *Docente del IES Lomo de la Herradura.*
- Hugo Fernández Ruíz: *Coordinador del Programa de Familias y Participación Educativa de la Dirección General de Ordenación e Innovación Educativa del Gobierno de Canarias.*
- Bryan del Cristo Pérez Ramírez: *Alumno de último curso del doble grado de Ingeniería Informática y Desarrollo de Videojuegos.*