



Universidad de
Oviedo



**ESCUELA POLITÉCNICA DE INGENIERÍA DE
GIJÓN.**

**GRADO EN INGENIERÍA INFORMÁTICA EN
TECNOLOGÍAS DE LA INFORMACIÓN**

ChronoStreetTurist

D. Bermúdez González, Ánder

TUTORA: D.^a González Aparicio, María Teresa

COTUTORES: D. Martín Liras, Luis

D. Rodríguez Vicente, Fernando

FECHA: Julio 2022

AGRADECIMIENTOS

En primer lugar, quiero agradecerles a mis padres Rubén y Nancy, a mi hermana Sandra y a mi novia Paula por el inmenso apoyo desde el primer día que decidí empezar mi etapa universitaria. En segundo lugar, a mi tutora académica Maite y cotutores Luis y Fernando de la empresa *HP SCDS* por haberme dado un soporte impresionante respondiendo a todas mis dudas diariamente y realizando revisiones quincenales. En tercer lugar, a mis amigos y compañeros de carrera que me han hecho la vida un poco más fácil y divertida.

Además, quiero agradecerme a mí mismo no solo por este proyecto, si no por el gran esfuerzo realizado estos últimos 4 años para poder ser una persona un poco más libre.

Por último, quiero agradecerte a ti, que por algún motivo tienes interés en mi trabajo y recordarás mi nombre un día más. Espero que mi proyecto te sirva de ayuda para cualesquiera sean tus motivaciones.

CONTENTS

1. Introduction	12
2. Goals and scope	13
2.1 GOALS.....	13
2.1.1 Mobile app.....	13
2.1.2 Content Management System.....	13
2.1.3 Integration with <i>Google Street View</i>	14
2.2 SCOPE.....	14
3. Previous studies and analyzes.....	15
3.1 <i>Archeoguide</i>	15
3.2 <i>Aurasma</i>	17
3.3 <i>Historical Photos in Augmented Reality</i>	17
3.4 COMPARATION.....	18
4. Planning	19
4.1 TASKS	19
4.2 GANTT DIAGRAM	22
4.3 USER STORIES DEFINITION	23
4.3.1 Develop first version of the app	23
4.3.2 Create <i>Vuforia</i> database.....	23
4.3.3 Develop cloud recognition.....	24
4.3.4 Create monument database.....	24
4.3.5 Develop user management.....	25
4.3.6 Create user basic frontend	26
4.3.7 Implement frontend monument upload	26
4.3.8 Implement manual monument upload in the app	27
4.3.9 Implement AR monument upload in the app	27

4.3.10 Stylize app	28
4.3.11 Implement monument visualization in frontend.....	28
4.3.12 Implement administrator tools.....	29
4.3.13 Implement monument geolocation	29
4.3.14 Investigate for alternatives.....	30
5. User requirements	31
5.1 FUNCTIONAL REQUIREMENTS.....	31
5.1.1 Mobile app.....	31
5.1.2 CMS.....	33
5.2 NON-FUNCTIONAL REQUIREMENTS.....	34
6. System requirements analysis.....	36
6.1 UPLOAD FORMS (SR-1)	36
6.2 REGISTER AND LOG IN (SR-2)	36
6.3 MOBILE APP INTERFACES (SR-3)	37
6.4 AR CAMERA (SR-4).....	37
6.5 INFORMATION TABS (SR-5)	38
6.6 USER MONUMENT MANAGEMENT (SR-6).....	38
6.7 ADMINISTRATOR MONUMENT MANAGEMENT (SR-7).....	39
7. Design.....	40
7.1 ARCHITECTURE.....	40
7.2 MONUMENTS	41
7.3 METADATA STRUCTURE	44
8. Screen prototypes	45
8.1 MOBILE APP	45
8.1.1 Icons	45
8.1.2 Screens.....	47
8.2 CMS.....	50

8.2.1 Menu	50
8.2.2 Image displayers	51
8.2.3 Screens	52
9. Testing	57
9.1 <i>Alpha</i> PHASE	57
9.2 TARGET TESTING	58
9.3 FORM VALIDATION	59
9.3.1 Manual upload form	59
9.3.2 AR upload form	62
9.3.3 Website upload form	62
10. Manuals	63
10.1 USER MANUAL	63
10.1.1 Scan a monument	63
10.1.2 Upload a monument (manual)	64
10.1.3 Upload a monument (AR)	66
10.1.4 Sign up on web service	68
10.1.5 Upload a monument from web service	69
10.1.6 Check my uploaded monuments	70
10.1.7 Log out	71
10.2 ADMINISTRATOR MANUAL	72
10.2.1 Approve monument	72
10.2.2 Delete monument	73
10.3 INSTALLER MANUAL	74
10.3.1 Import unity project	74
10.3.2 Set up Vuforia license	75
10.3.3 Set up Vuforia database	78
11. Technology research	81

Figure 8.19.- Administrator top menu	50
Figure 8.20.- Administrator side menu.....	51
Figure 8.21.- User top menu.....	51
Figure 8.22.- User side menu.....	51
Figure 8.23.- User image displayer	52
Figure 8.24.- Administrator image displayer.....	52
Figure 8.25.- Login page	53
Figure 8.26.- Registration page	53
Figures 8.27. and 8.28.- User and password recovery page	54
Figure 8.29.-My monuments page.....	54
Figure 8.30.- Add monument page.....	55
Figure 8.31.- Index page.....	55
Figure 8.32.- Approve monuments tab.....	56
Figure 8.33.- Project web service from a OnePlus 8	56
Figure 9.2.- AR camera bug on Blackview BV9700 Pro	57
Figure 9.2.- Vuforia target rating.....	59
Figures 9.3. and 9.4.- Username validation errors.....	60
Figures 9.5. and 9.6.- Image selector validation errors	60
Figures 9.7. and 9.8.- URL validation errors.....	61
Figures 9.9. and 9.10.- Upload form correctly validated.....	62
Figure 10.1.- Scan monument indicated button.....	63
Figures 10.2. and 10.3.- Monument scanning	64
Figures 10.4. and 10.5.- Instructions to manually upload a monument	64
Figure 10.6.- Manual upload form example	65
Figures 10.7. and 10.8.- Instructions to upload a monument using AR	66
Figure 10.9.- AR upload form example.....	67
Figure 10.10.- AR upload camera	67

11.1	<i>Vuforia</i>	81
11.2	APP DEVELOPMENT PLATFORM	81
11.3	CONTENT MANAGER SYSTEM (CMS)	82
11.3.1	<i>ConvertForms</i>	83
11.3.2	<i>Sigplus</i>	84
11.4	HOSTING.....	84
11.5	REPOSITORY	85
11.6	DATABASE.....	85
11.7	TEXT EDITORS	86
11.8	FTP CLIENT	87
11.9	DOCUMENTATION	87
12.	Budget.....	88
12.1	SOFTWARE.....	88
12.2	HARDWARE AND OTHERS	89
12.3	HUMAN RESOURCES	90
12.4	TOTAL	90
13.	Proposed improvements	91
13.1	<i>Google Street View</i>	91
13.1.1	<i>Bing StreetSide View</i>	91
13.1.2	Mapillary	91
13.2	USER LOGIN FROM THE APP	92
13.3	INCREASE BUDGET	92
13.4	USABILITY	92
14.	Conclusions	93
15.	Glossary	94
16.	Bibliography	96

FIGURES

Figure 1.1.- Example of Augmented Reality in phones (Pokemon GO)..... 12

Figure 3.1.- Original image used to test Archeoguide..... 16

Figure 3.2.- Result of applying Archeoguide to Figure 3.1..... 16

Figure 3.3.- Image superposition using “Historical Photos in Augmented Reality” 18

Figure 4.1.- Gantt diagram 22

Figure 4.2.- Monument entity on CMS database 25

Figure 4.3.- CMS database architecture at this point 25

Figure 4.4.- CMS database architecture adding state attribute..... 28

Figure 7.1.- General architecture of the project..... 40

Figure 7.2.- Mobile app navigation flow 41

Figure 7.3.- State diagram of a monument 42

Figure 7.4.- Flow of monuments among the service 43

Figure 7.5.- Metadata example..... 44

Figure 8.1.- Home icon..... 45

Figure 8.2.- Exit icon..... 45

Figure 8.3.- Information icon 46

Figure 8.4.- Screenshot icon 46

Figures 8.5. and 8.6.- AR state icon in its two different states..... 46

Figure 8.7.- Monument information icon 46

Figure 8.8.- Icon to take photos on AR monument upload 47

Figure 8.9.- App main screen 47

Figures 8.10. and 8.11.- Information screens 48

Figure 8.12.- Monument upload selector..... 48

Figures 8.13. and 8.14.- AR and manual monument upload screens 49

Figures 8.15. and 8.16.- Scan monument camera interface..... 49

Figures 8.17. and 8.18.- AR monument upload camera interface..... 50

Figures 10.11. and 10.12.- Steps to access to the web service from the mobile app..... 68

Figure 10.13.- Sign up button..... 68

Figure 10.14.- Sign up form example..... 69

Figure 10.15.- Log in form example 69

Figure 10.16.- Web upload form example..... 70

Figure 10.17.- My monuments tab 71

Figure 10.18.- Index tab 71

Figure 10.19.- Approve monuments tab..... 72

Figure 10.20.- Approved monument 73

Figure 10.21.- Deleted monument..... 73

Figure 10.22.- Project repository..... 74

Figure 10.23.- Unity Hub 75

Figure 10.24.- Vuforia developer portal..... 75

Figure 10.25.- License manager 76

Figure 10.26.- License key 76

Figure 10.27.- AR Camera element..... 77

Figure 10.28.- AR Camera properties 77

Figure 10.29.- Vuforia engine conf. 78

Figure 10.30.- Target manager. 78

Figure 10.31.- Vuforia database creator. 79

Figure 10.32.- Database access keys. 79

Figure 10.33.- Cloud recognition element..... 80

Figure 10.34.- Unity database access keys. 80

Figure 10.35.- Joomla.php server access keys..... 80

Figure 11.1.- Vuforia logo..... 81

Figure 11.2.- Unity logo 82

Figure 11.3.- Joomla logo..... 83

Figure 11.4.- CloudAccess logo	84
Figure 11.5.- Gitlab logo	85
Figures 11.6. and 11.7- Visual Studio and Notepad++ logos.....	86
Figure 11.8.- WinSCP logo	87
Figure 11.9.- Microsoft office	87

TABLES

Table 4.1.- List of tasks	21
Table 9.1.- Target testing on OnePlus 8	58
Table 9.2.- Target testing on Xiaomi Redmi Note 9 Pro.....	58
Table 12.1.- Software budget	88
Table 12.2.- Hardware budget	89
Table 12.3.- Human resources budget	90
Table 12.4.- Final budget.....	90

1. Introduction

Augmented Reality is the set of technologies that make possible the interaction between objects in the physical and virtual world through technological devices such as smartphones, cameras, tablets, computers, etc. It is important to differentiate AR[3] from virtual reality, which tries to generate a digital environment and does not interact with the real world.

During the recent years, Augmented Reality has become very important due to its fast evolution, allowing its integration in many daily devices like phones, cars, cameras, etc.



Figure 1.1.- Example of Augmented Reality in phones (Pokemon GO)

ChronoStreetTurist is an example of an Augmented Reality application. It allows us to visualize the evolution of different monuments and verify *in-situ* how the passage of time has affected them.

Keywords:

ChronoStreetTurist, *Vuforia*, Mobile app, Web service, Augmented Reality, AR, Cloud recognition, *Unity*, C#, *Joomla*, PHP, CMS, Monuments, Metadata, *ConvertForms*, *Sigplus*.

Repository: [EPI-ChronoStreetTurist.rar](#)

2. Goals and scope

2.1 GOALS

The project is divided into different phases, each of them with different objectives.

1. Mobile app
2. Content Management System
3. Integration with *Google Street View*

2.1.1 Mobile app

Mobile app is the main important part of the project. The goals are as follows:

1. Develop an Augmented Reality application that allows images to be overlapped on different monuments of the city.
2. Activate cloud recognition[4] to achieve a unique and centralized database for every user.
3. Allow user to upload its own monuments[17] to the database using AR or manually.
4. Store monument's metadata when user uploads a monument.

2.1.2 Content Management System

CMS[5] is the backbone of the project, and its objectives have the following clear and important purposes:

1. Create a web service in which users can see the monuments they have uploaded and their status.
2. Allow user to manually upload monuments also from the website.
3. Create a way for administrators to approve or delete monuments uploaded by users.

2.1.3 Integration with *Google Street View*

Google Street View is a technology featured in *Google Maps* and *Google Earth* that provides interactive panoramas from positions along many streets in the world. The goal is to investigate how to integrate current *Google Street View* photos with the photos uploaded by the user.

2.2 SCOPE

The scope of our project is principally limited by the budget. It will be essential to use free resources in each phase which will greatly limit things like database capacity, web performance, premium functionalities of many plugins used, etc. In addition to this, the licenses of the plugins for both the mobile application and the CMS must be free of charge and free to use (MIT[\[16\]](#) or GPL[\[10\]](#) licenses are the ones preferred).

3. Previous studies and analyzes

In this section, other projects similar to *ChronoStreetTurist* are described. Then, a comparison between them is done.

3.1 Archeoguide

Archeoguide is a project in which the user manages to visualize, by merging the real with the virtual world, 3D reconstructions of places in ruins (as can be seen in Fig 3.1 and Fig 3.2). The technique used by the service is based on the tracking position using image techniques. Next, a method that compares the similarities between the images collected by the camera and those found in the database whose associated coordinates are closed to those provided by the mobile device's GPS is used. If two images match in at least a 30%, the process of overlaying the image in the database over the one provided by the device will start. The techniques used in the process of correlation between two images are: translation, rotation of images so that they have the same perspective and scale of the base image.



Figure 3.1.- Original image used to test Archeoguide



Figure 3.2.- Result of applying Archeoguide to Figure 3.1.

3.2 *Aurasma*

Aurasma was a mobile app for *Android* and *IOS* (deleted from stores on 1st July 2019) oriented towards educational purposes. It allowed to combine real-world objects, images, and places with digital elements (animations, videos or other images). When the user focuses the object from the real world, the digital element comes out allowing interaction with the environment.

The application was based on scenes, which could be created by any user through the mobile application itself or with the online tool. The steps to carry out the process of adding a scene was as follows:

1. Register in the official page of *Aurasma*.
2. Create a new Augmented Reality scene (*aura*) and upload the images that will work as bookmarks.
3. Upload the corresponding digital element with that image.
4. Post the scene to a new channel.

3.3 *Historical Photos in Augmented Reality*

This app makes use of Augmented Reality to show historical images of monuments of the city of Helsinki. The service is based on two visual recognition techniques which are as follows:

1. Point cloud: this technique consists in performing a 3D reconstruction of the monument by making a huge number of photos of it in different weather conditions, different lighting, days and hours. This feature facilitates to place manually images in certain position and perspective (as can be seen in Fig. 3.3).
2. 2D images as markers: like in *ChronoStreetTurist*, this application makes use of monument pictures to achieve a superposition of historical images when focusing on that monument. In addition, by using the GPS sensors of the mobile phone itself, this application provides a modest 3D tracking in case of visual tracking method failures.



Figure 3.3.- Image superposition using “Historical Photos in Augmented Reality”

3.4 COMPARATION

Each of the services mentioned above have strong and weak points compared to our project.

Aurasma, for example, allows animations and video uploading while *ChronoStreetTurist* only allows images. *Historical Photos in Augmented Reality* is more powerful than our service since it allows 3D reconstruction of the monument, improving the comparison of the image with the monument and *Archeoguide* uses geolocation instead of service camera. This method avoids tracking problems caused by weather, illumination, etc.

ChronoStreetTurist allows the user to upload its own monuments by two different ways. Moreover, since it uses the *Vuforia SDK* [20], it only needs an image of the actual monument to achieve superposition, which is an easier solution. Finally, unlike previous services, *ChronoStreetTurist* offers an additional web service with user management allowing them to manage its own monuments.

4. Planning

The agile methodology *scrum* has been used throughout the project. The work has been divided into 5 sprints with different durations and a variable number of user stories. Sprints 1 and 3 were focused on the app, while sprints 2 and 4 were focused on CMS. Sprint 5 was focused on looking into geolocation and *Google Street View* integration. Each sprint is self-conclusive, obtaining at the end of each a functional version of the service.

The meetings of this methodology have taken place every two weeks involving student and tutors in each of them. At the end of the testing phase, a demonstration has been made.

4.1 TASKS

Id	Name	Duration (days)	Start date	End date	Predecessors
1	Requirements specification	5	10/01/22	15/01/22	
1.1	Functional requirements	3	10/01/22	12/01/22	
1.2	Non-functional requirements	2	13/01/22	14/01/22	1.1
2	Previous analyzes	29	14/01/22	11/02/22	
2.1	Unity and C# learning	7	14/01/22	20/01/22	
2.2	CMS Basics	8	04/02/22	11/02/22	
3	Sprint 1 (app)	15	28/01/22	11/02/22	1, 2.1

3.1	Develop first version of the app	9	28/01/22	05/02/22	
3.2	Create Vuforia database	3	03/02/22	05/02/22	
3.3	Develop cloud recognition	6	06/02/22	11/02/22	3.1, 3.2
4	Sprint 2 (CMS)	28	12/02/22	11/03/22	2.2, 3
4.1	Create monument database	10	12/02/22	21/02/22	
4.2	Develop user management	10	22/02/22	03/03/22	4.1
4.3	Create user basic frontend	3	04/03/22	06/03/22	4.2
4.4	Implement frontend monument upload	5	07/03/22	11/03/22	4.3
5	Sprint 3 (app)	28	12/03/22	08/04/22	4
5.1	Implement manual monument upload on app	15	12/03/22	26/03/22	
5.2	Implement AR monument upload on app	10	27/03/22	05/04/22	5.1
5.3	Stylize app	3	27/03/22	05/04/22	5.2
6	Sprint 4 (CMS)	28	09/04/22	06/05/22	5
6.1	Implement monument visualization in frontend	7	09/04/22	15/04/22	

6.2	Implement admin tools	21	16/04/22	06/05/22	6.1
7	Sprint 5 (Google StreetView)	20	07/05/22	26/05/22	6
7.1	Implement monument geolocalization	5	07/05/22	11/05/22	
7.2	Investigate alternatives	15	12/05/22	26/05/22	
8	Testing	20	27/05/22	15/06/22	7
8.1	App testing	2	27/05/22	28/05/22	
8.2	App bug fixes	12	29/05/22	09/06/22	8.1
8.3	CMS testing	1	10/06/22	10/06/22	8.2
8.4	CMS bug fixes	5	11/06/22	15/06/22	8.3
9	Documentation	25	16/06/22	10/07/22	8

Table 4.1.- List of tasks

4.2 GANTT DIAGRAM

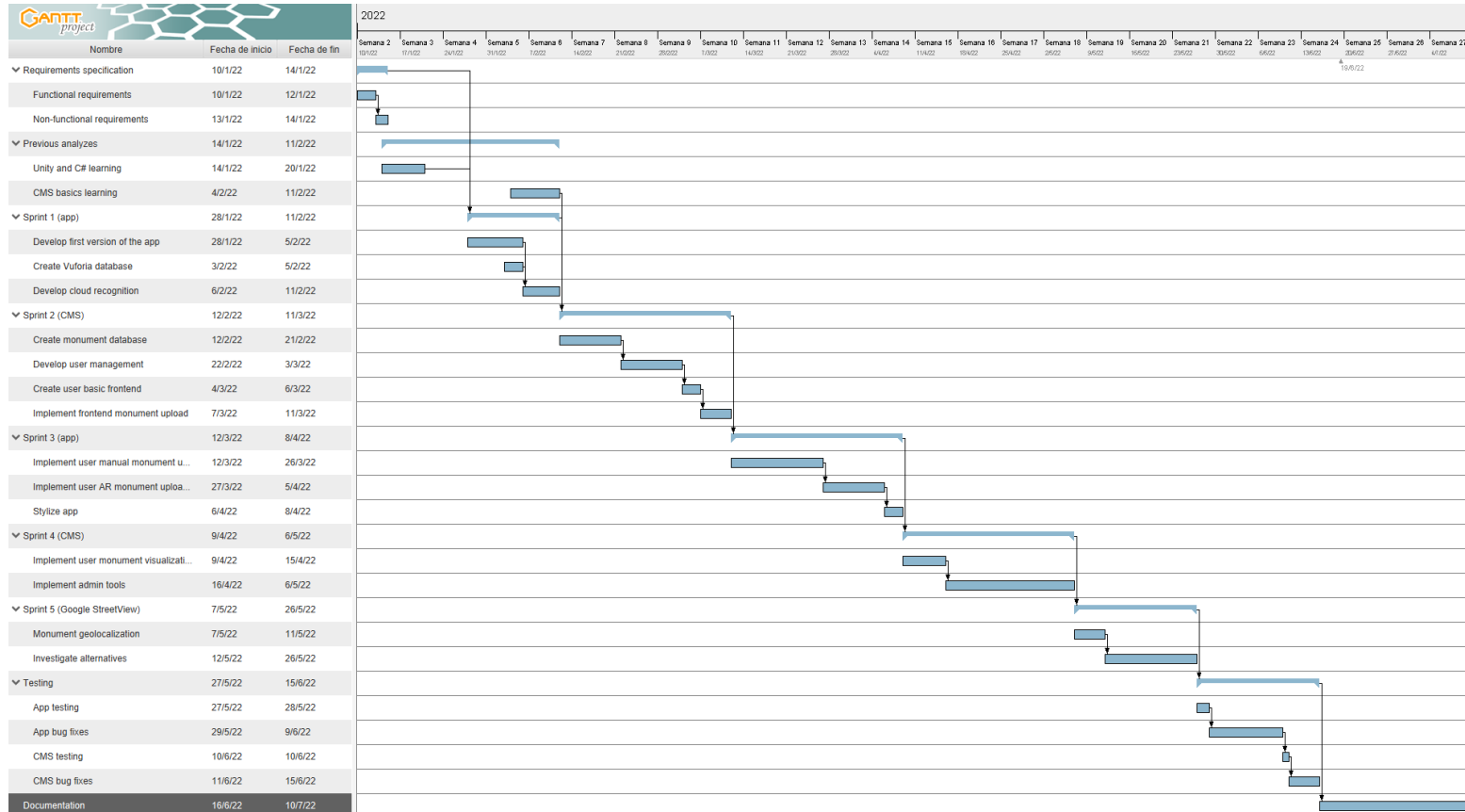


Figure 4.1.- Gantt diagram

4.3 USER STORIES DEFINITION

In this section, user stories will be defined along with their acceptance criteria.

4.3.1 Develop first version of the app

As a developer, I want to have a basic prototype of the final mobile application.

Acceptance criteria:

- Simple interface.
- Exit button.
- HP and *Uniovi* logos at the top.
- AR works with hardcoded monuments.
- When a monument is recognized, the info button will appear to give user more information about it.

4.3.2 Create *Vuforia* database

As a developer, I want a Vuforia database[\[29\]](#) to store monument images along with its metadata.

Acceptance criteria:

- Create database on *Vuforia* webpage.
- Locate database keys to later upload images remotely.

4.3.3 Develop cloud recognition

As a developer, I want to use *Vuforia* database to store the monuments used by the mobile app.

Acceptance criteria:

- Establish connection between *Vuforia* database and mobile app.
- Recognize monuments stored in the database.
- Instantly notify any change made in the database to the app.

4.3.4 Create monument database

As a developer, I want another database to store all the monument data.

Detailed description:

To make mobile app work it is needed a current photo of the monument, an old photo of the same monument and its metadata (position, scale, info, etc). *Vuforia* database is only able to store the current photo and metadata. So, the proposed solution is to store in metadata a direct URL[\[25\]](#) to the old photo. For this reason, another database to get that link is needed.

Acceptance criteria:

- Create database with sharable links to its elements.
- Create an entity called monument with the following attributes.

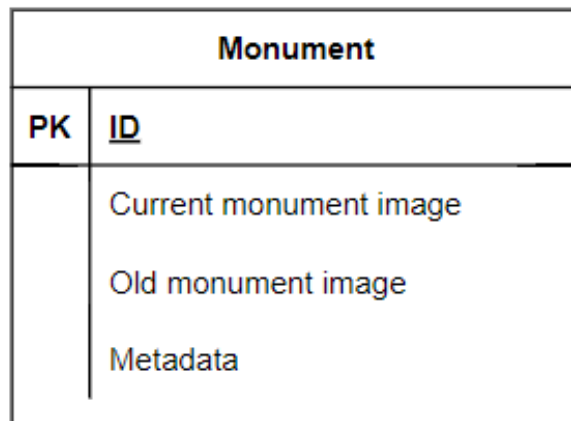


Figure 4.2.- Monument entity on CMS database

4.3.5 Develop user management

As a developer I want to have a user management.

Acceptance criteria:

- Differentiate between users.
- Include user entity in the database.

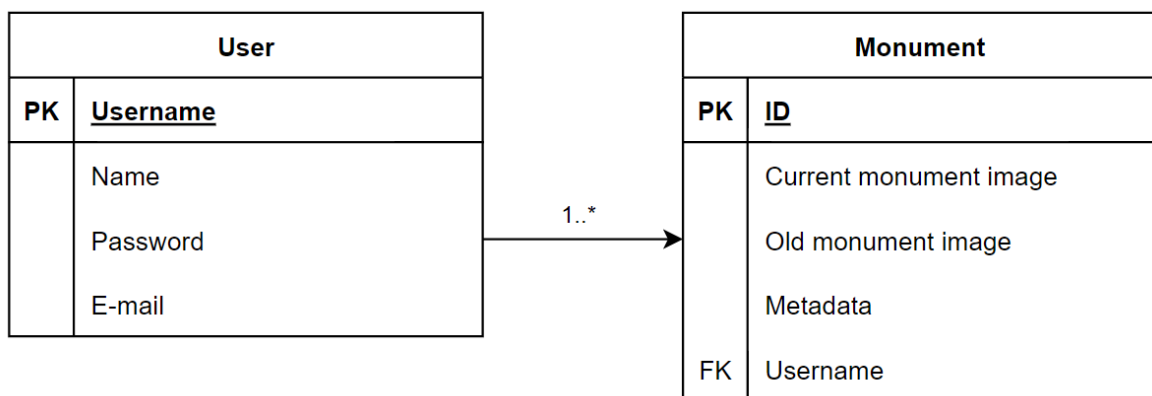


Figure 4.3.- CMS database architecture at this point

4.3.6 Create user basic frontend

As a developer, I want to create a small web service on the CMS for users.

Acceptance criteria:

- Register and login page.
- Store users correctly.
- Display welcome page when user is logged (“Welcome [username]”).

4.3.7 Implement frontend monument upload

As a developer, I want users to be able to upload monuments to CMS database [\[6\]](#) through the web service.

Acceptance criteria:

- Upload form.
- Image uploader for current and old monument image.
- Fields to fill monument metadata.
- Compulsory fields.
- Optional fields. In this case, they will be replaced by default data.
- Send button to store monument in database in case form is correct.
- Store monument in CMS database.

4.3.8 Implement manual monument upload in the app

As a developer, I want users to be able to upload monuments to CMS database through the mobile app.

Acceptance criteria:

- Upload form.
- Image uploader for current and old monument image.
- Fields to fill monument metadata.
- Compulsory fields.
- Optional fields. In this case, they will be replaced by default data.
- *Send* button to store monument in database in case form is correct.
- Store monument in CMS database.

4.3.9 Implement AR monument upload in the app

As a developer, I want users to be able to upload monuments to CMS database using AR.

Detailed description:

The proposal is that if you have an old photo of the monument and you are in front of it (in the same position), AR will recognize it when you point it with your mobile camera getting the current image of it and some metadata (position and scale).

Acceptance criteria:

- Upload form.
- Image uploader for old monument image.
- Fields to fill some monument metadata.
- Open camera button. It will check if the form is correctly completed and will open phone camera.
- Display button to take a photo if monument is recognized by AR using the old image uploaded by the user.

- Make button invisible if monument is lost.
- Store monument in CMS database.

4.3.10 Stylize app

As a developer, I want an attractive and easy to use interface.

Acceptance criteria:

- Attractive interface approved by testers.
- Ease of access to all features.

4.3.11 Implement monument visualization in frontend

As a developer, I want users to be able to see its uploaded monuments in the web service.

Acceptance criteria:

- Mechanism to recognize the user who uploads a monument from the app.
- Check by logged user its monument images.
- Divide monuments into *pending* and *approved*. If a monument is *approved*, it means that it will be in the app. If it is *pending*, it means the administrators have not yet cataloged it. By default, an uploaded monument will be in *pending* state.

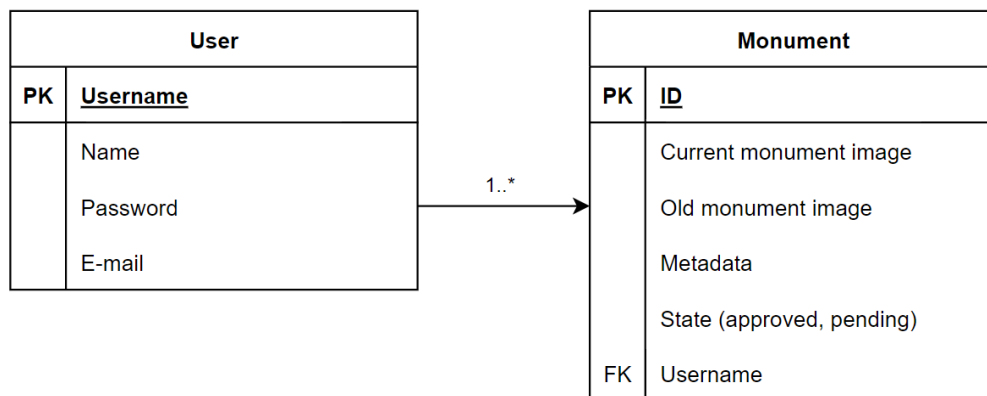


Figure 4.4.- CMS database architecture adding state attribute

4.3.12 Implement administrator tools

As a developer, I want administrators to be able to approve or delete a monument uploaded by a user.

Acceptance criteria:

- Admins will have an additional tab on CMS in which pending monuments will be displayed.
- Administrators can approve a monument with just clicking one button.
- Also, they can delete the monument if there is a problem with it (wrong metadata, incorrect images, etc).
- Automatically upload monument to *Vuforia* database when admin approves it.

4.3.13 Implement monument geolocation

As a developer, I want to the coordinates (altitude, longitude and latitude) of a monument when it is uploaded using AR.

Acceptance criteria:

- Have and additional element in metadata when a monument is uploaded by mobile App using AR called *location* which will be a 3-dimension array composed by latitude, longitude and altitude of the image taken.
- Request image location if it is not enabled.

4.3.14 Investigate for alternatives

As a developer, I want to investigate how I can integrate located monuments into *Google Street View* or similar services.

Acceptance criteria:

- Study viability of this integration.
- Search for free alternatives.

5. User requirements

User requirements are classified into functional and non-functional.

5.1 FUNCTIONAL REQUIREMENTS

Functional requirements are product features or functions that developers must implement to enable users to accomplish their tasks. They are enumerated in UFR-X^[23] format.

5.1.1 Mobile app

1. Interface.
 - 1.1. *Uniovi* and *HP* logos must be visible (UFR-1).
 - 1.2. Exit button must be visible on the principal screen (UFR-2).
 - 1.3. The rest of the screens must show a return button (UFR-3).
2. Monument scan.
 - 2.1. Users must be able to point the mobile camera at a monument and visualize the following data.
 - 2.1.1. Old monument image superposed (UFR-4).
 - 2.1.2. Information button to know more about the building (UFR-5).
 - 2.2. Mobile app will let users activate or deactivate AR in real time. (UFR-6).
 - 2.3. Mobile app will let users take photos (UFR-7).
3. Monument upload.
 - 3.1. Users must be able to upload monuments in any of the following ways.
 - 3.1.1. Manual upload.
 - 3.1.1.1. Mobile app will request users the following information (UFR-8).

- 3.1.1.1.1. Username.
- 3.1.1.1.2. Current monument photo.
- 3.1.1.1.3. Old monument photo.
- 3.1.1.1.4. Position of the monument.
- 3.1.1.1.5. Scale of the old image.
- 3.1.1.1.6. URL with monument information.

3.1.2. AR upload.

3.1.2.1. Mobile app will request users the following information (UFR-9).

- 3.1.2.1.1. Username.
- 3.1.2.1.2. Old monument photo.
- 3.1.2.1.3. URL with monument information.

3.1.2.2. Mobile app must let user take a photo of the monument in real time (UFR-10).

3.2. Monument upload can be done anonymously if user wishes (UFR-11).

3.3. Mobile app will show a button to clean all fields of the form (UFR-12).

3.4. Paths of the images loaded in the form must be viewable by the user (UFR-13).

4. Information.

4.1. Mobile app must have an explanation of the following concepts (UFR-14).

- 4.1.1. Manual upload.
- 4.1.2. AR upload.

4.2. Mobile app must have a tab with the following elements (UFR-15).

- 4.2.1. Brief presentation of the developer.
- 4.2.2. Objectives and functionalities of the project.
- 4.2.3. *Uniovi* and HP citation.
- 4.2.4. Direct access to the web service.

5. Form validation.

5.1. Error messages will specify the field in which the error is found (UFR-16).

5.2. Image loaders will only accept common image formats (*gifs* not allowed) (UFR-17).

5.3. A Confirmation message will appear when the monument is successfully uploaded into the CMS database (UFR-18).

5.1.2 CMS

1. User registration.
 - 1.1. Guessers must be able to register in the system providing the following information (UFR-19).
 - 1.1.1. Real name.
 - 1.1.2. Username.
 - 1.1.3. Password.
 - 1.1.4. E-mail.
 - 1.2. Username must be unique (UFR-20).
 - 1.3. Password must be confirmed by guesser (UFR-21).
2. User login.
 - 2.1. Users must be able to login into the system providing the following information (UFR-22).
 - 2.1.1. Username.
 - 2.1.2. Password.
 - 2.2. User must have a mechanism to remember his password in case of loss (UFR-23).
3. Monument upload.
 - 3.1. User can upload monuments through the web service providing the following data (UFR-24).
 - 3.1.1. Current monument photo.
 - 3.1.2. Old monument photo.
 - 3.1.3. Position of the monument.
 - 3.1.4. Scale of the old image.
 - 3.1.5. URL with monument information.
 - 3.2. Additionally, web service will collect the username of the user who upload the monument (UFR-25).

4. Monument visualization.
 - 4.1. Users must be able to visualize monuments uploaded from the app or web (UFR-26).
 - 4.2. Users will only be able to see their uploaded monuments (UFR-27).
 - 4.3. There will be a differentiation between *pending* and *approved* monuments (UFR-28).

5. Administrator tools.
 - 5.1. Users with admin role will be able to perform any user action (UFR-29).
 - 5.2. Admins will have an additional tab with the following elements.
 - 5.2.1. *Pending* monuments uploaded by any user (UFR-30).
 - 5.2.2. Button to approve a monument (UFR-31).
 - 5.2.3. Button to delete a monument (UFR-32).

5.2 NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements define system attributes such as security, reliability, scalability or usability. They are enumerated in UNFR-X[24] format.

1. Interface.
 - 1.1. Clarity and ease of use (UNFR-1).
 - 1.1.1. Bright colors.
 - 1.1.2. Margins and separation between elements.
 - 1.1.3. Clear font.
 - 1.1.4. Large icons.
 - 1.2. Feedback.
 - 1.2.1. Buttons must give instant response when pressed (shaded, color changes, animations, etc) (UNFR-2).
 - 1.2.2. Dynamic loading messages (UNFR-3).
2. Form validation.
 - 2.1. It will be performed dynamically by the app/web service when possible. If that is not possible, it will perform a query to the database. (UNFR-4).

3. Security.

3.1. User passwords must be encrypted using MD5 algorithm (UNFR-5).

3.2. Development must avoid request user for sensitive information (UNFR-6).

6. System requirements analysis

System requirements are configurations that a system must have in order for a hardware or software application to run smoothly and efficiently. They are associated with one or more user requirements enumerated in SR-X[21] format.

6.1 UPLOAD FORMS (SR-1)

Upload forms are the way proposed to ask user for monument data. Different upload forms are created with its corresponding elements. A checkbox is present in mobile app forms. If it is marked, username will not be required, and monument will be anonymously uploaded.

SR-1 covers UFR-8, UFR-9, UFR-11, UFR-12, UFR-13, UFR-16, UFR-17, UFR-18, UFR-24, UFR-25, UNFR-3, UNFR-4 and UNFR-6.

Acceptance criteria:

- URLs must start by *http://* or *https://* and end in a TLD[22].
- Position and scale must be integer or decimal numbers.
- Image uploaders must have a label which shows the path of the image loaded.

6.2 REGISTER AND LOG IN (SR-2)

Service uses a user management system to distinguish between users and admins. For this reason, a user register and log in is needed.

SR-2 covers UFR-19, UFR-20, UFR-21, UFR-22, UFR-23, UFR-24 and UNFR-5.

Acceptance criteria:

- From website main screen, a guesser must have a button to register.
- To do that, guesser must give its name, email, username and a password in a registration form.
- Once registered, guesser become a user able to login with its credentials.
- From website main screen, a user must have an option to log in by typing its username and password in 2 different fields.
- From website main screen, a user must have a button to remember its password by typing its username or email and checking mailbox.

6.3 MOBILE APP INTERFACES (SR-3)

SR-3 covers UFR-1, UFR-2, UFR-3, UNFR-1 and UNFR-2.

Acceptance criteria:

- User must be able to return to the previous screen using a button located in a screen corner.
- User must be able to exit from the app with a button ubicated in a screen corner if he is in the main screen.
- Logos from *Uniovi* and HP must be shown in the top among the app.

6.4 AR CAMERA (SR-4)

SR-4 covers UFR-4, UFR-4, UFR-6 and UFR-10.

Acceptance criteria:

- User must be able to enable and disable AR using a toggle button in a corner of the camera interface.
- User must be able to take photos using a button in another corner of the camera interface.

- User must be able to scan monuments and see the superposed photo in front of the monument and its information by pressing a button in the top of the camera interface.

6.5 INFORMATION TABS (SR-5)

Although the user has more detailed manuals at his disposal, app will summarize some of its contents.

SR-5 covers UFR-14, UFR-15 and UNFR-1.

Acceptance criteria:

- User must be able to read the author's name and surnames, goals and scope of the service.
- User must be able to read the differences between manual and AR uploads.

6.6 USER MONUMENT MANAGEMENT (SR-6)

Monuments are stored in a relational database and for this reason users would be able to perform some actions with them.

SR-6 covers UFR-26, USFR-27 and UFR-28.

Acceptance criteria:

- User must be able check monuments uploaded by him bot from the app and website.
- User must be able to distinguish between pending and approved monuments.

6.7 ADMINISTRATOR MONUMENT MANAGEMENT (SR-7)

SR-7 covers UFR-29, USFR-30, UFR-31 and UFR-32.

Acceptance criteria:

- Administrators must be able to approve or delete a pending monument.

7. Design

In this section, the proposed architecture of the project is described. Then, possible states that a monument can have along with the possible routes that they can take are explained. At the end, metadata design is also described.

7.1 ARCHITECTURE

The project is divided into 3 main platforms: mobile app, CMS and Vuforia. They are connected to each other using different mechanisms as we can see in Fig. 7.1.

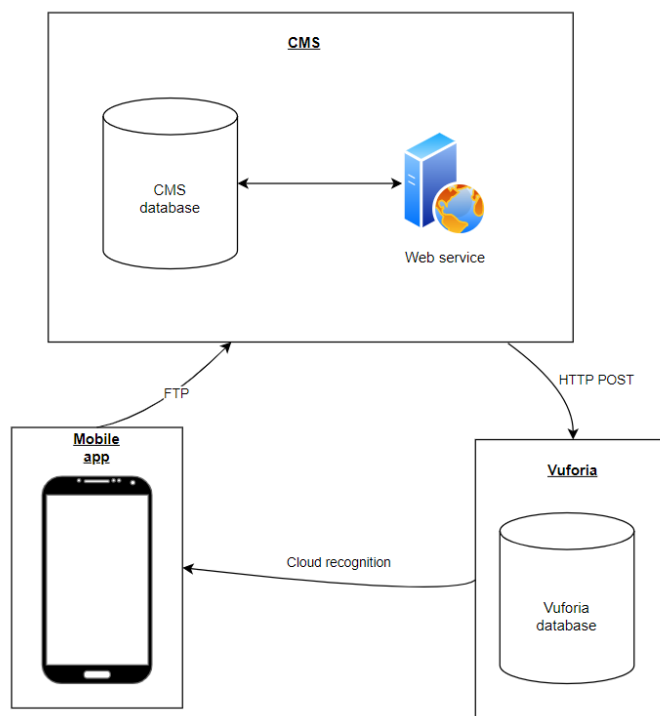


Figure 7.1.- General architecture of the project

Mobile application has several screens with different actions associated such as uploading or scanning monuments. Fig 7.2 shows the navigation flow of the app.

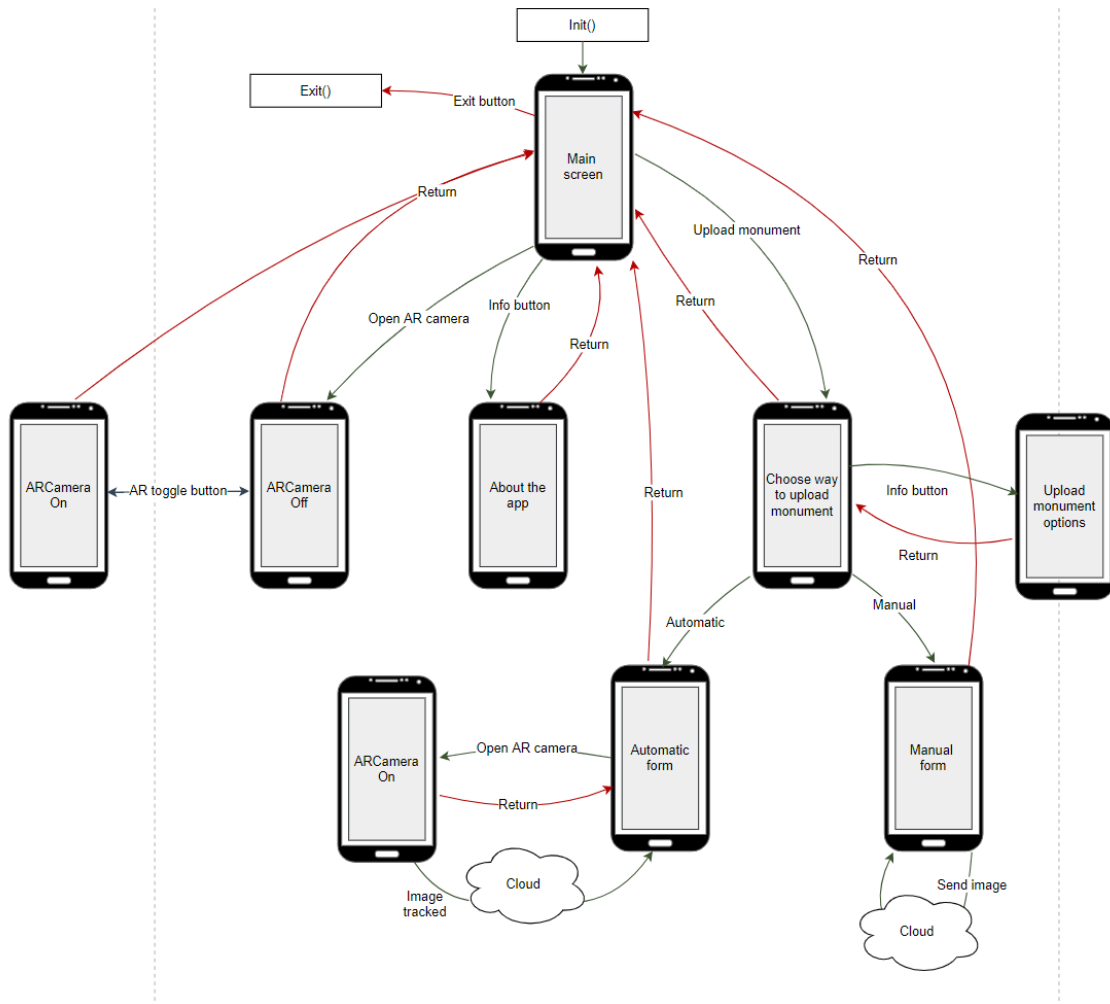


Figure 7.2.- Mobile app navigation flow

7.2 MONUMENTS

Once a monument is uploaded, it is assigned the status of pending. In this state, monument will be only visible on the web by admins and by the user who uploaded it. If an administrator approves it, it will become part of *Vuforia* database, and therefore the mobile application. If monument contains errors such as metadata failures, wrong image format, etc it will be deleted from database by administrators.

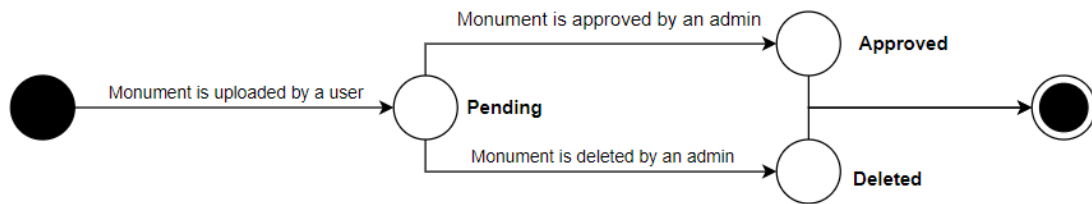


Figure 7.3.- State diagram of a monument

For the sake of a better understanding of the monument's behavior, Fig 7.4 includes a brief explanation of every step a monument does in the service.

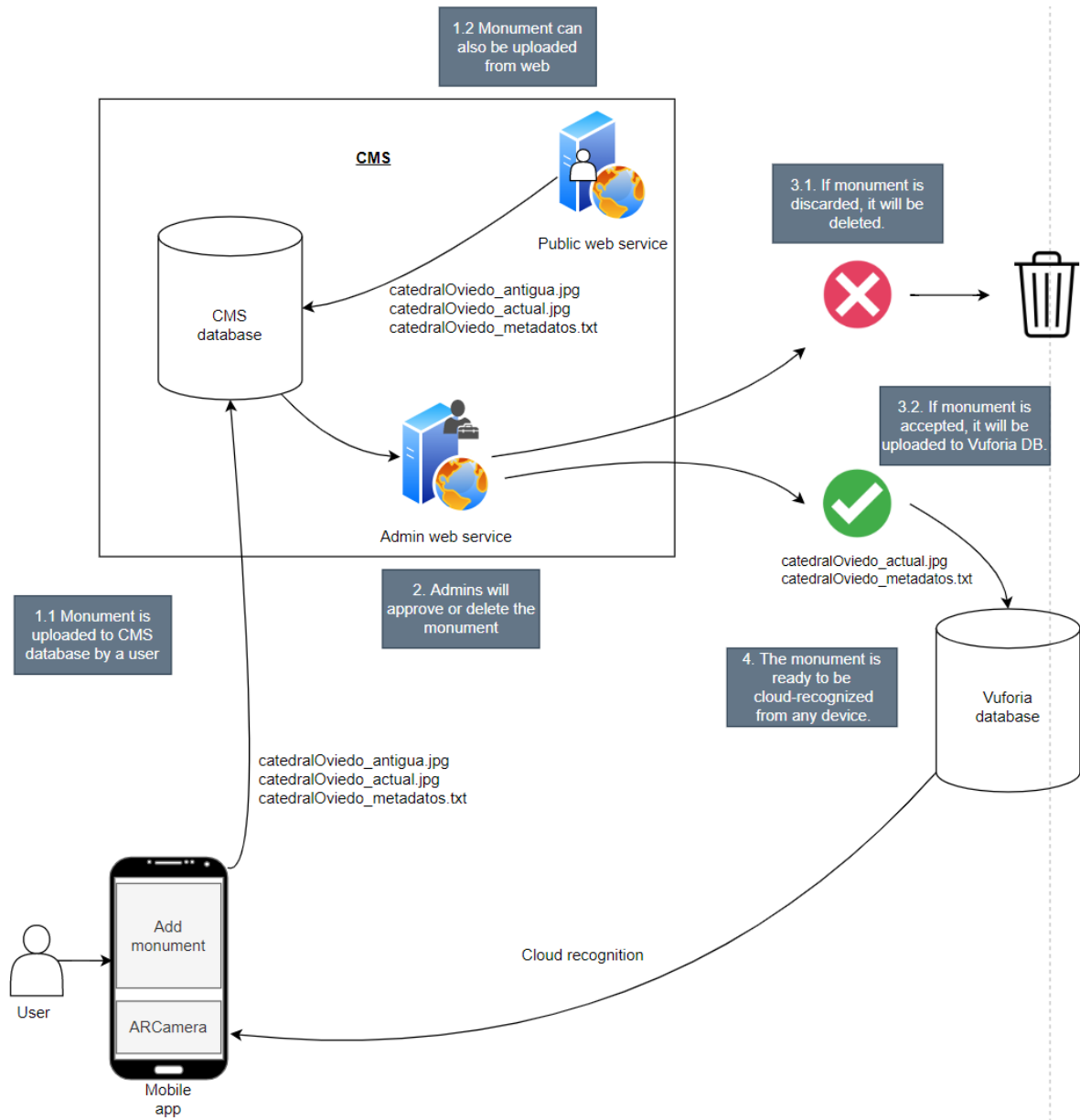


Figure 7.4.- Flow of monuments among the service

7.3 METADATA STRUCTURE

Each monument has an associated metadata that will be essential to achieve superposition. This file is in *txt* format because *Vuforia* DB[7] only accepts plain text as metadata but its content it will be structured as a JSON[14]. The elements are the following:

- **Position:** it is an array with the x, y and z coordinates where the old image has to be placed relative to the current image. In Fig 7.5, it will be in the middle (0, 0, 0).
- **URL:** it is a direct URL in which we can find the old monument image ready to be downloaded.
- **Info:** it is a URL in which we can find more information about the monument. When it is recognized by the app. A button will take us to the aforementioned.
- **Scale:** it is an array with height, width and depth of the old image relative to the current image. In fig 7.5, old image needs to be reduced to a 10% of its width and height. Depth is not used in this version because we don't support 3D.
- **Location:** it is an array with latitude, longitude, and altitude of the monument. In Fig 7.5 all elements are settled to 0 because the app did not have *GPS* permissions.

```
{
  "position":["0","0","0"],
  "url":"https://chronostreetturist.joomla.com/images/Monumentos/anderbggo/Aprobados/03-06-2022 12:10/antigua.jpg",
  "info":"https://google.es",
  "scale":["0.1","0.1","0.15"],
  "location":["0","0","0"]
}
```

Figure 7.5.- Metadata example

8. Screen prototypes

In this section, screen prototypes used for the project are exposed. The proposal is a simple design with as few elements as possible in each screen.

8.1 MOBILE APP

Its design maintains blue tones to match HP color palette and pretends to be intuitive since all elements are consistent between screens. Now, icons and screens are listed with a brief explanation.

8.1.1 Icons

Fig 8.1 shows the icon used to return to the previous screen. This is found on all screens except the main one.



Figure 8.1.- Home icon

Fig 8.2 shows the icon used to exit from the app. It is found on main screen.



Figure 8.2.- Exit icon

Fig 8.3 shows the icon used to get more information. It is used on selected screens.



Figure 8.3.- Information icon

Fig 8.4 shows the icon to take a screenshot. This icon is exclusive to the camera interface.

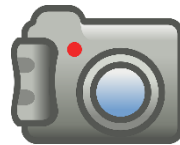


Figure 8.4.- Screenshot icon

Figs 8.5 and 8.6 shows the same icon in its two possible states. Green means that AR is turned on while red means the opposite.



Figures 8.5. and 8.6.- AR state icon in its two different states

Fig 8.7 shows the icon to open the URL with information about the monument tracked. Icon appears when a monument is tracked, and its design is different to the other information button (Fig 8.3) to indicate that its meaning is different.



Figure 8.7.- Monument information icon

To finish the list, Fig 8.8 shows the icon used by the camera interface of the AR upload to take a photo when a monument is tracked. Icon only appears if the old monument photo allows camera to track the actual monument when user points it.



Figure 8.8.- Icon to take photos on AR monument upload

8.1.2 Screens

Fig 8.9 shows the principal screen with the main functionalities: scan and upload a monument.

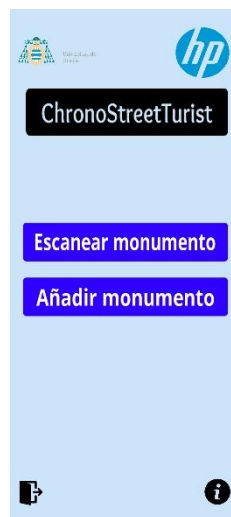
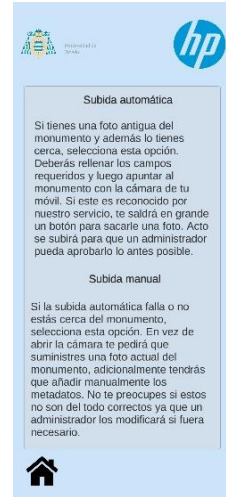
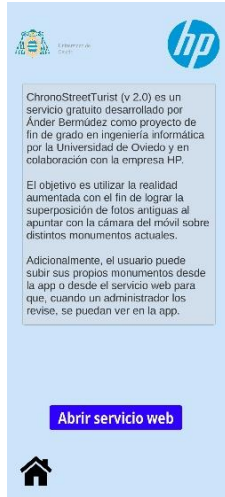


Figure 8.9.- App main screen

Figs 8.10 and 8.11 show the two information screens present in the app. Fig 8.10 gives us a brief explanation of the project and its objective while 8.11 defines the two different upload methods implemented.



Figures 8.10. and 8.11.- Information screens

Fig 8.12 shows the screen used to select between manual and AR upload (*Automático*).

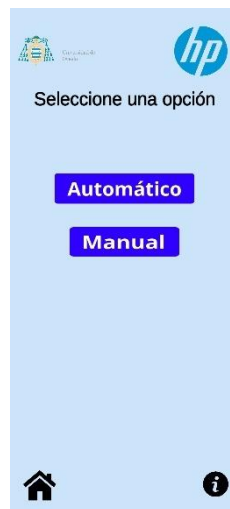
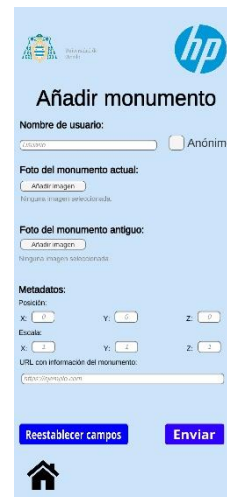


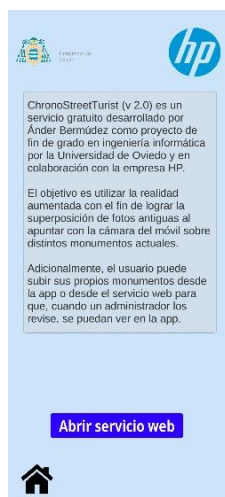
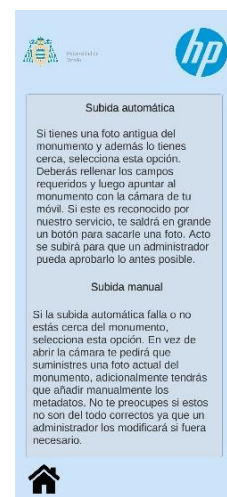
Figure 8.12.- Monument upload selector

Figs 8.12 and 8.13 shows the screens used to upload monuments. As we can see, manual uploads require to complete more fields.

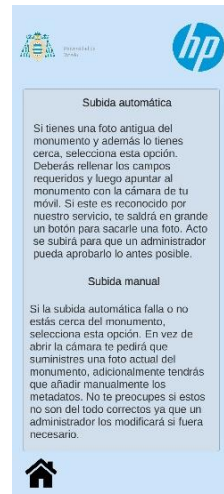
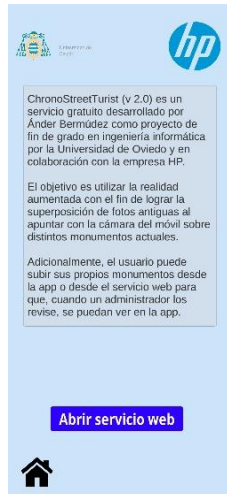
Figures 8.13. and 8.14.- AR and manual monument upload screens

Camera interface used to scan monuments is showed in Figs 8.15 and 8.16. As it can be seen, when monument is tracked and AR is activated (green color), the photo of the old monument appears along with the information button.

Figures 8.15. and 8.16.- Scan monument camera interface

Figs 8.17 and 8.18 show camera interface when monument is uploaded using AR. In this case, when the monument is tracked, button to take a photo appears.



Figures 8.17. and 8.18.- AR monument upload camera interface

8.2 CMS

Web service included in CMS has a simple and uncluttered design. It mainly uses the colors black and blue.

8.2.1 Menu

Web navigation works thanks to the top and side menus. These menus are different depending on whether the user has administrator permissions or not.



Figure 8.19.- Administrator top menu



Figure 8.20.- Administrator side menu



Figure 8.21.- User top menu



Figure 8.22.- User side menu

8.2.2 Image displayers

There are two different types of image displayers. One is used to allow the user to see their monuments in more detail and the other to allow administrators approve or delete monuments uploaded.

As can be seen in Fig 8.23, user image displayers have 3 buttons. Arrow buttons are used to navigate through the pictures while X button is used to close the image displayer.



Figure 8.23.- User image displayer

Fig 8.24 shows an example of an administrator image displayer. Arrow buttons are used for the same as the previous example, but X button takes a different meaning: delete the complete monument. Moreover, floppy disk button is used to approve the monument and camera button doesn't have a particular use now, but it is intended for future versions to show monument metadata.

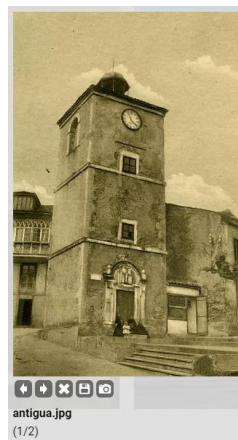


Figure 8.24.- Administrator image displayer

8.2.3 Screens

Fig 8.25 shows login page design of the web service.

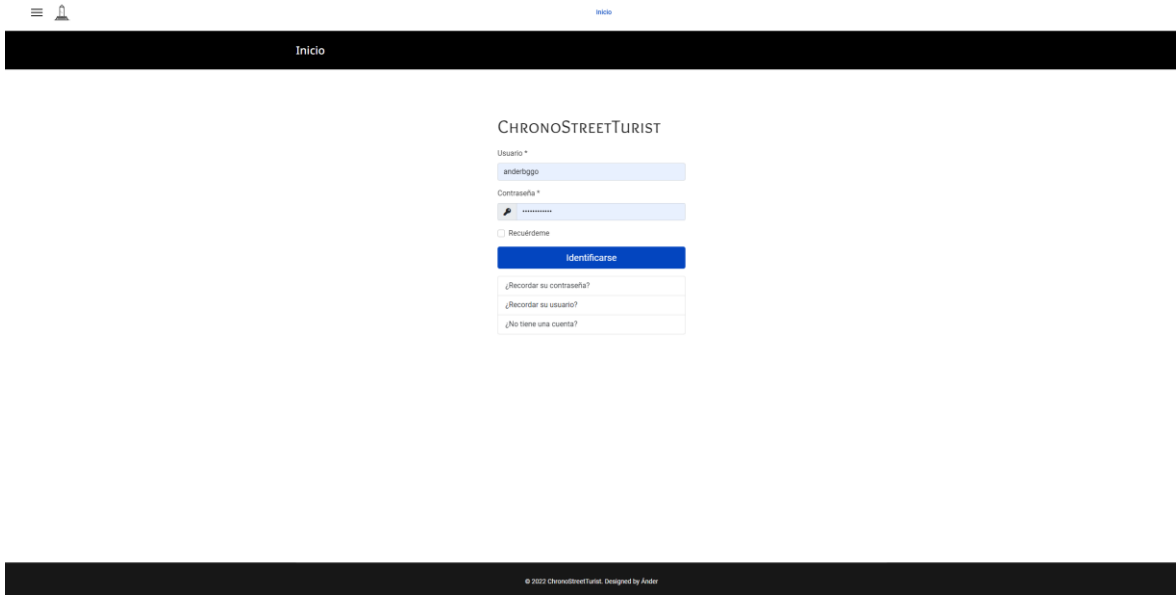


Figure 8.25.- Login page

To make a registration, user needs a name, username, e-mail, and password as can be seen in Fig 8.26.

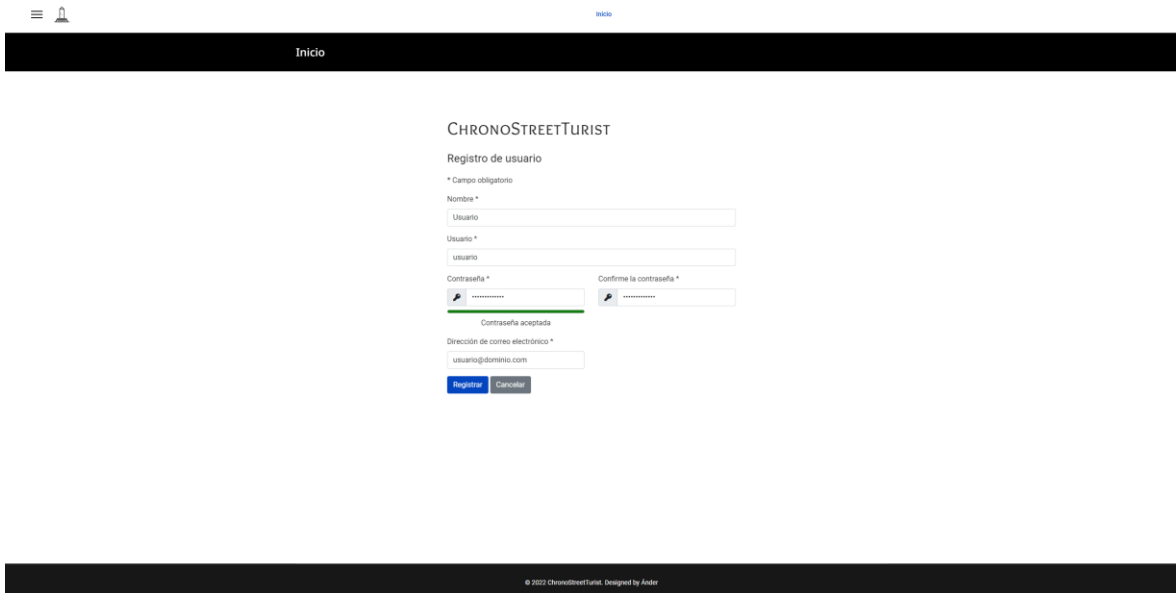


Figure 8.26.- Registration page

If user forgets its user or password. It can be recovered thanks to its email.

CHRONOSTREETTURIST

Por favor, introduzca la dirección de correo electrónico que tiene asociada con su cuenta de usuario. Su usuario le será enviado si la dirección que nos facilita es la correcta.

Dirección de correo electrónico *

Enviar

CHRONOSTREETTURIST

Por favor, introduzca la dirección de correo electrónico para su cuenta. Se le enviará un código de verificación. Una vez que lo haya recibido, podrá seleccionar una nueva contraseña para su cuenta.

Dirección de correo electrónico *

Enviar

Figures 8.27. and 8.28.- User and password recovery page

Once logged, user can check its own monuments in the corresponding tab. There are two different user image displays: one for approved monuments and the other for pending ones as it can be seen on figure 8.29.

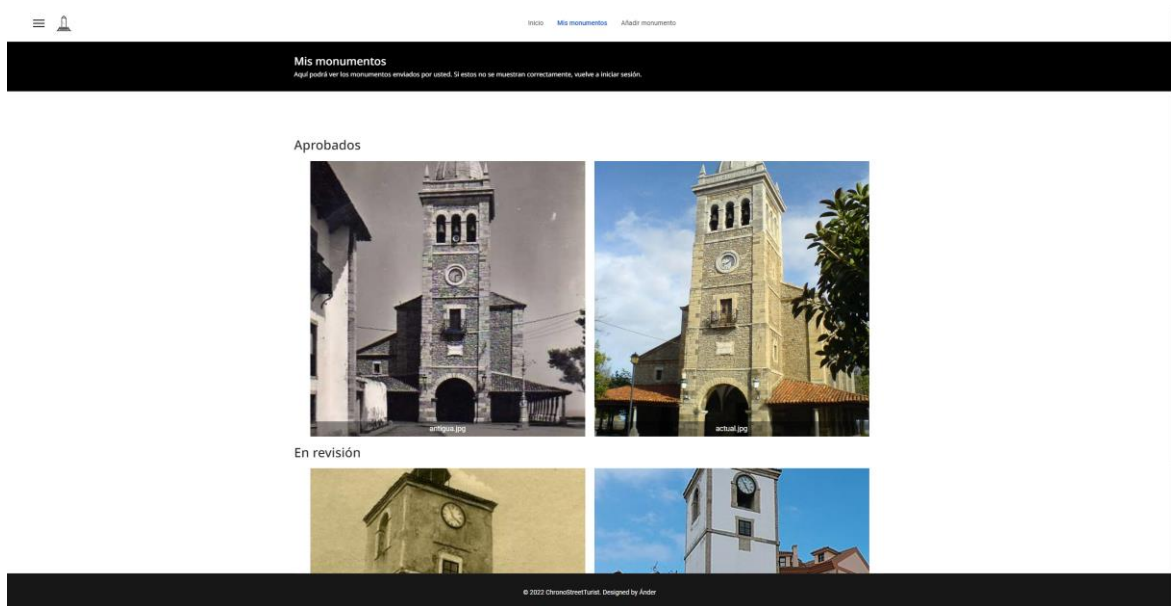


Figure 8.29.-My monuments page

Users can also upload photos from the web service. Fig 8.30 shows the corresponding upload form.

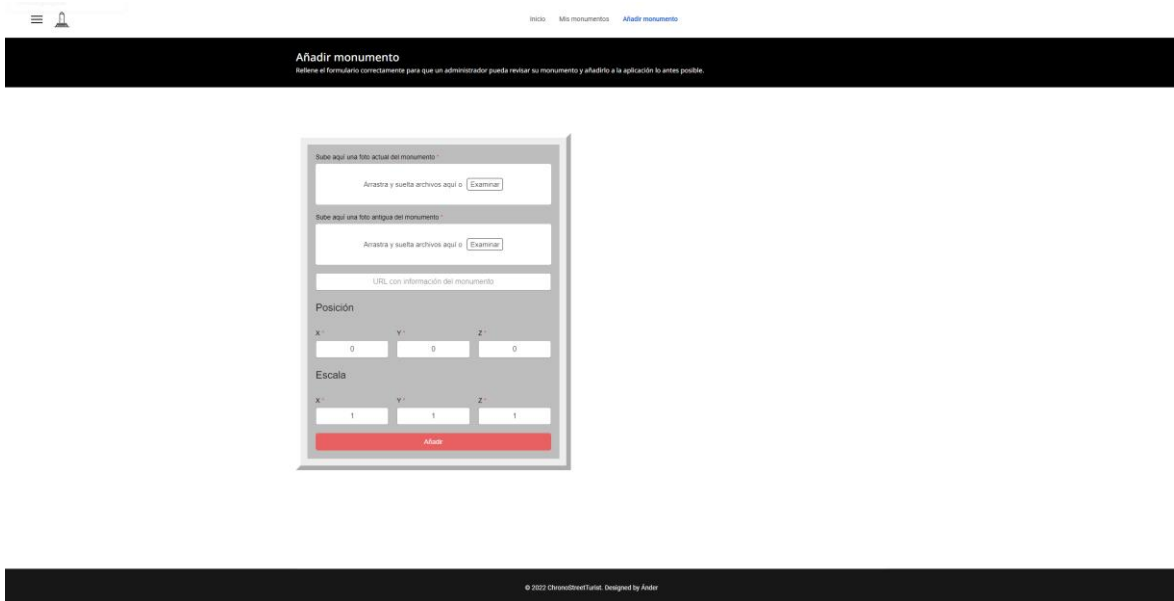


Figure 8.30.- Add monument page

Index tab is used to log out.

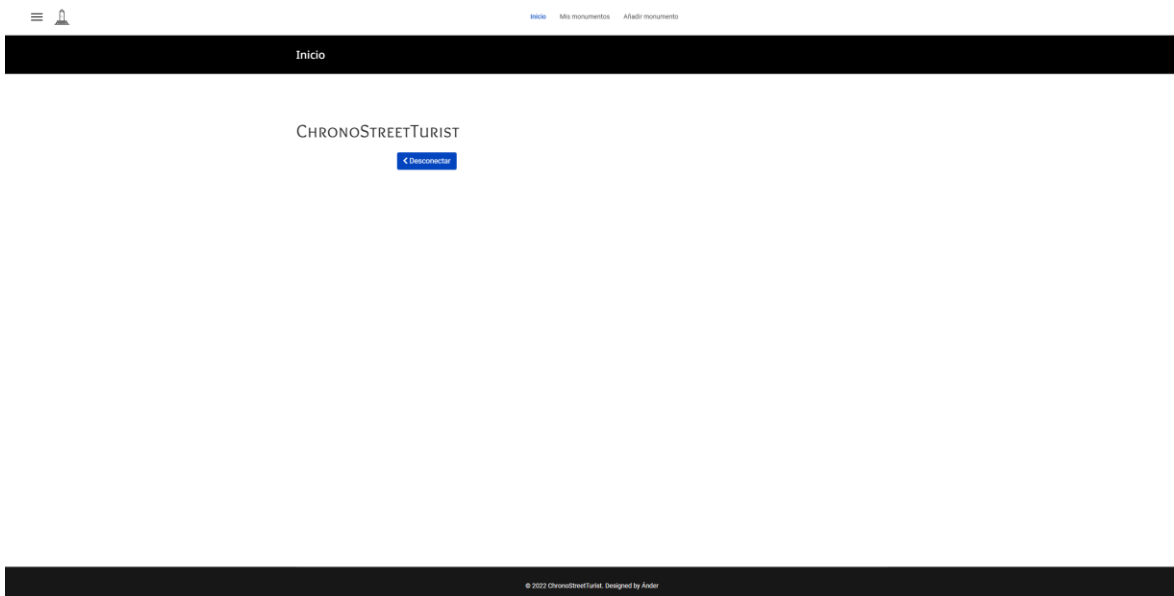


Figure 8.31.- Index page

Approve monuments is an exclusive tab for administrators. They can see pending approval monuments here. Clicking on any of the two images, the image displayer of Fig 8.24 will appear.

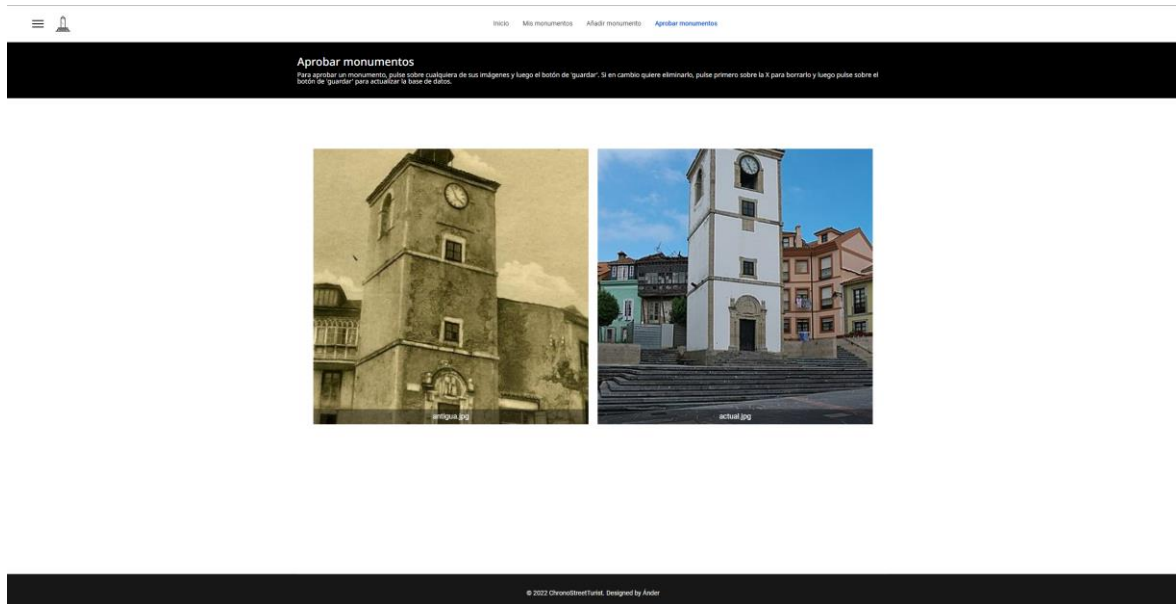


Figure 8.32.- Approve monuments tab

Web service is responsive. An example of this can be seen on Fig 8.33.



Figure 8.33.- Project web service from a OnePlus 8

9. Testing

One of the most important stages in the development of a software product is testing. In the case of *ChronoStreetTurist*, it becomes even more important since it involves different product communicated between them. Any not fully tested use case could be fatal for the databases or web service.

9.1 Alpha PHASE

During the *Alpha* phase, APK[2] of the app was distributed among a small group of persons with *Android* devices. Any failure experienced by any of these people would be communicated to the developer.

During this phase, AR camera issues with *Blackview BV9700 Pro* phone were detected as we can see in Fig 9.1. This bug has low priority and has not been fixed yet.



Figure 9.2.- AR camera bug on Blackview BV9700 Pro

9.2 TARGET TESTING

It consists of doing scan tests on different targets (*Luanco* church and *Luanco* clock tower) that are already in the *Vuforia* database and check service tracking capacity. These tests were performed at different times of the day with different lighting. In addition, they were carried out with different mobile devices.

Target testing on <i>OnePlus 8</i>				
#	Day hour (aprox.)	Illumination	Target	Is it correctly recognized?
1	11:00	High	Church	Yes
2	11:00	High	Clock tower	Yes
3	20:00	Medium	Church	Yes
4	20:00	Medium	Clock tower	Yes
5	23:00	Low	Church	Yes
6	23:00	Low	Clock tower	Sometimes

Table 9.1.- Target testing on *OnePlus 8*

Target testing on <i>Xiaomi Redmi Note 9 Pro</i>				
#	Day hour (aprox.)	Illumination	Target	Is it correctly recognized?
1	11:00	High	Church	Yes
2	11:00	High	Clock tower	Yes
3	20:00	Medium	Church	Yes
4	20:00	Medium	Clock tower	Yes
5	23:00	Low	Church	Yes
6	23:00	Low	Clock tower	No

Table 9.2.- Target testing on *Xiaomi Redmi Note 9 Pro*

As we can see in the previous tables, target recognition works moderately well. Good lighting conditions ensures that scan will work correctly while poor lighting conditions can cause recognition problems.

However, the quality of the image is also relevant. Church's photo was taken with a better camera than clock tower's photo and for this reason church target has better rating[18].




<input type="checkbox"/>	Target Name	Rating 
<input type="checkbox"/>	 Iglesia	★★★★★
<input type="checkbox"/>	 Torre_reloj	★★★★☆

Figure 9.2.- Vuforia target rating

9.3 FORM VALIDATION

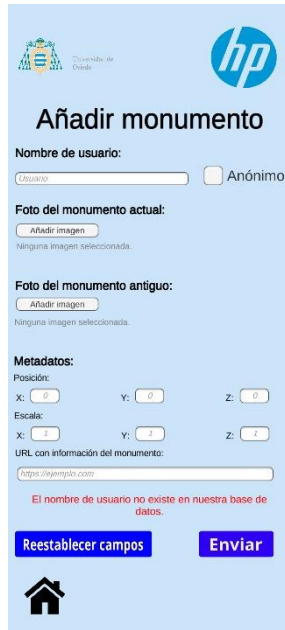
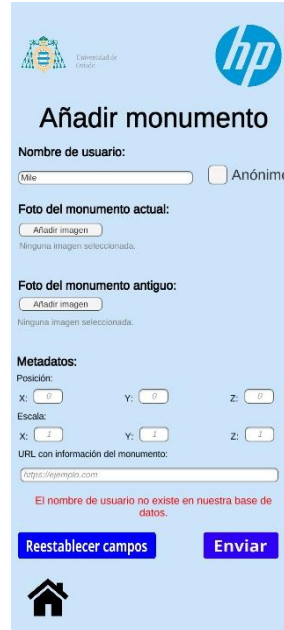
Validation ensures the data consistency. Several tests were done on the forms to avoid any validation failure and all of them has passed successfully.

Instead of using a table to show each test, it has been preferred to group them into different small sections with to make a better reading. In addition, each test depends on the previous one since the validation errors are in order.

9.3.1 Manual upload form

Test group 1: Username

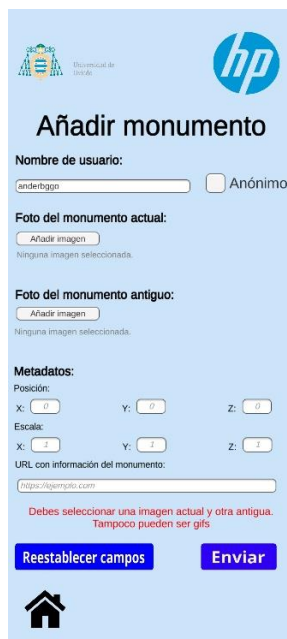
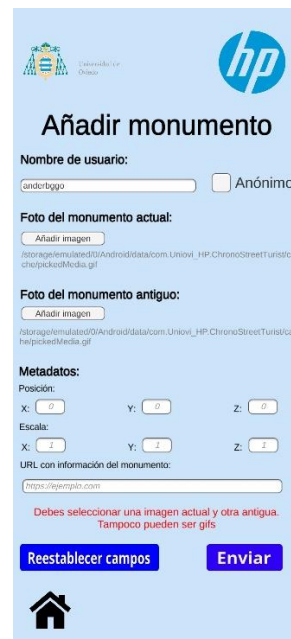
When username is not typed or it doesn't exist in CMS, the expected output is a validation error as we can see in Figs 9.3 and 9.4.

Figures 9.3. and 9.4.- Username validation errors

Test group 2: Image selectors

If any of the image selectors is not used or image format is invalid, expected output is the one showed on Figs 9.5 and 9.6.

Figures 9.5. and 9.6.- Image selector validation errors

Test group 3: Position and Scale

These fields are optional and only accept numbers and a point or a comma. For that reason, they will never give us a validation error. The use of the point or the comma to separate the decimals is indifferent.

Test group 4: URL

If URL is not typed or has an invalid format. Expected output can be seen of Figs 9.7 and 9.8.



The screenshot shows the 'Añadir monumento' form with the following fields and values:

- Nombre de usuario: Anónimo
- Foto del monumento actual:
- Foto del monumento antiguo:
- Metadatos:
 - Posición: X: Y: Z:
 - Escala: X: Y: Z:
- URL con información del monumento:

Below the URL field, there is a red error message: "Introduce una URL válida". At the bottom, there are two buttons: "Reestablecer campos" and "Enviar".



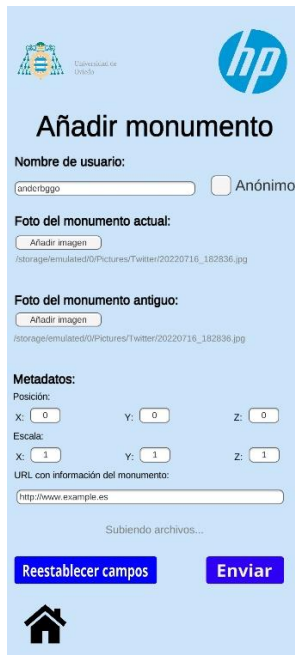
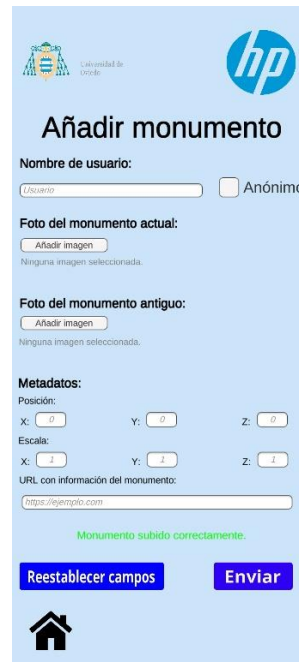
The screenshot shows the 'Añadir monumento' form with the following fields and values:

- Nombre de usuario: Anónimo
- Foto del monumento actual:
- Foto del monumento antiguo:
- Metadatos:
 - Posición: X: Y: Z:
 - Escala: X: Y: Z:
- URL con información del monumento:

Below the URL field, there is a red error message: "Introduce una URL válida". At the bottom, there are two buttons: "Reestablecer campos" and "Enviar".

Figures 9.7. and 9.8.- URL validation errors

Otherwise, monument will be correctly uploaded.

Figures 9.9. and 9.10.- Upload form correctly validated

9.3.2 AR upload form

Group tests 1, 2 and 4 were performed to check all possible test combinations. To avoid redundancy, they will not be explained again.

9.3.3 Website upload form

Group tests 2, 3 and 4 were performed to check all possible test combinations.

10. Manuals

In this section are the manuals for each role. Installers are those who are in charge of preparing the service to function at first. They will also be in charge of maintaining the service if it is necessary.

10.1 USER MANUAL

Below are the instructions for a user to be able to use all the functionalities of the project.

10.1.1 Scan a monument

To scan a monument, user needs to have installed the mobile app on an Android device. Then, click on the button indicated in Fig 10.1.

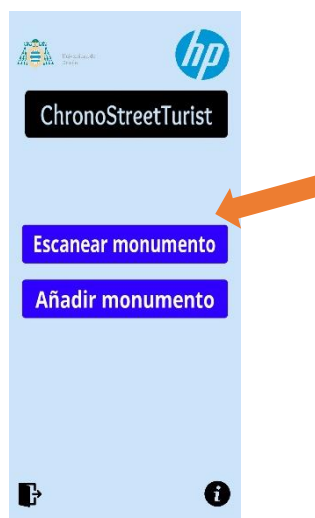


Figure 10.1.- Scan monument indicated button

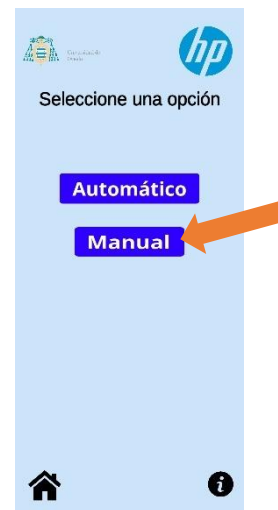
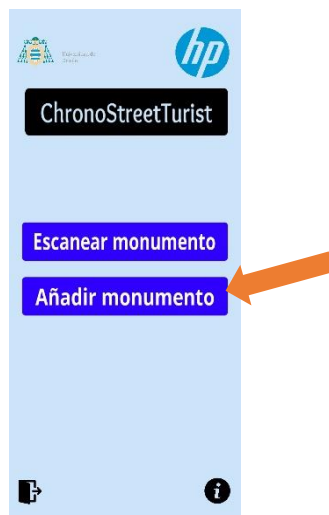
Finally, user needs to check that AR is activated and point the camera to the monument.



Figures 10.2. and 10.3.- Monument scanning

10.1.2 Upload a monument (manual)

User needs to have Android app installed and follow the instructions of Figs 10.4 and 10.5.



Figures 10.4. and 10.5.- Instructions to manually upload a monument

Now, user fills the form with the required data. It will be important to consider the following requirements:

- Username must be registered on the web service. If you do not have one, you can create it as will be explained later in this section or upload it anonymously by checking the corresponding checkbox.
- Current and old monument photos can be in any of the most common image formats except .gif.
- All fields from metadata must be integer or decimal numbers. If they are left blank, they will be filled with default data.
- URL must be, obviously, an URL (start from *http://* or *https://* and end in an *TLD*).

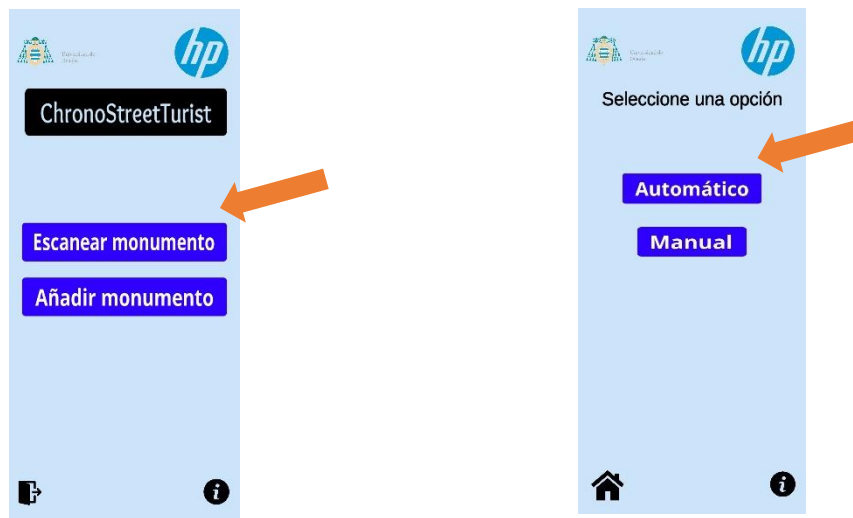
If any of the requirements is not met when you press the send button, an error message will indicate the problem. Otherwise, a message will inform you that the monument has been correctly uploaded.



Figure 10.6.- Manual upload form example

10.1.3 Upload a monument (AR)

User needs to have Android app installed and follow the instructions of Figs 10.7 and 10.8.



Figures 10.7. and 10.8.- Instructions to upload a monument using AR

Now, user fills the form with the required data. It will be important to consider the following requirements:

- Username must be registered on the web service. If you do not have one, you can create it as will be explained later in this section or upload it anonymously by checking the corresponding checkbox.
- Old monument photos can be in any of the most common image formats except .gif.
- URL must be, obviously, an URL (start from *http://* or *https://* and end in an *TLD*).



The screenshot shows a mobile application interface for adding a monument. At the top, there are logos for the University of Oviedo and HP. The main heading is "Añadir monumento". Below this, there is a "Nombre de usuario:" label with a text input field containing "Usuario" and a checked "Anónimo" checkbox. The next section is "Foto del monumento antiguo:" with a "Añadir imagen" button and a small thumbnail image of a monument. Below that is a "URL con información del monumento:" label with a text input field containing "http://www.ejemplo.com". At the bottom of the form are two blue buttons: "Reestablecer campos" and "Abrir cámara". A home icon is located at the very bottom of the screen.

Figure 10.9.- AR upload form example

If any of the requirements is not met when you press “Abrir cámara” button, an error message will indicate the problem. Otherwise, camera will open to take a photo of the current monument as we can see in Fig 10.10.



Figure 10.10.- AR upload camera

Once user takes a photo of the monument, it will be automatically uploaded to the database.

10.1.4 Sign up on web service

User must know the web URL. If not, user can access from the application by following the steps of Figs 10.11 and 10.12.



Figures 10.11. and 10.12.- Steps to access to the web service from the mobile app

Once there, user must click on button indicated in Fig 10.13.

CHRONOSTREETTURIST

Usuario *

Por favor, complete este campo

Contraseña *

Por favor, complete este campo

Recuérdeme

Identificarse

¿Recordar su contraseña?

¿Recordar su usuario?

¿No tiene una cuenta?

Figure 10.13.- Sign up button

Finally, user must fill the sign-up button following the requirements of each field and click on “Registrar” button.

CHRONOSTREETTURIST

Registro de usuario

* Campo obligatorio

Nombre *

Usuario *

Contraseña *

Confirme la contraseña *

Contraseña aceptada

Dirección de correo electrónico *

Figure 10.14.- Sign up form example

10.1.5 Upload a monument from web service

User must know the web URL and be registered as explained in the previous section. Then, user will log in as showed in Fig 10.15.

CHRONOSTREETTURIST

Usuario *

Contraseña *

Recuérdeme

¿Recordar su contraseña?

¿Recordar su usuario?

¿No tiene una cuenta?

Figure 10.15.- Log in form example

Once logged, user must use top or side menu to enter on “Añadir monumento” tab and fill the form following the same requisites of section 10.1.2.



The form contains the following fields and elements:

- Section: 'Sube aquí una foto actual del monumento'
- File upload area: 'Arrastra y suelta archivos aquí o Examinar'
- Uploaded file: 'iglesia_actual.jpg 0.4 MB x'
- Section: 'Sube aquí una foto antigua del monumento'
- File upload area: 'Arrastra y suelta archivos aquí o Examinar'
- Uploaded file: 'iglesia_antigua.jpg 0.1 MB x'
- URL field: 'https://www.ecample.com'
- Section: 'Posición'
- Coordinates: X: 0.2, Y: 5, Z: 0.87
- Section: 'Escala'
- Scale values: X: 3, Y: 0.1, Z: 1
- Red 'Añadir' button at the bottom.

Figure 10.16.- Web upload form example

10.1.6 Check my uploaded monuments

User must follow the same steps as previous section but this time the selected menu item will be “Mis monumentos”.

Inicio **Mis monumentos** Añadir monumento

Mis monumentos

Aquí podrá ver los monumentos enviados por usted. Si estos no se muestran correctamente, vuelve a iniciar sesión.

Aprobados



En revisión



Figure 10.17.- My monuments tab

10.1.7 Log out

User must follow steps from Fig 10.18.

Inicio **Mis monumentos** Añadir monumento

Inicio

CHRONOSTREETTURIST

← Desconectar

Figure 10.18.- Index tab

10.2 ADMINISTRATOR MANUAL

Administrators are normal users with privileges to approve or delete monuments.

10.2.1 Approve monument

Administrator must log in on web service as explained on previous section and click on “Aprobar monumentos” tab. Then, he must click on any of the photos of the monument.

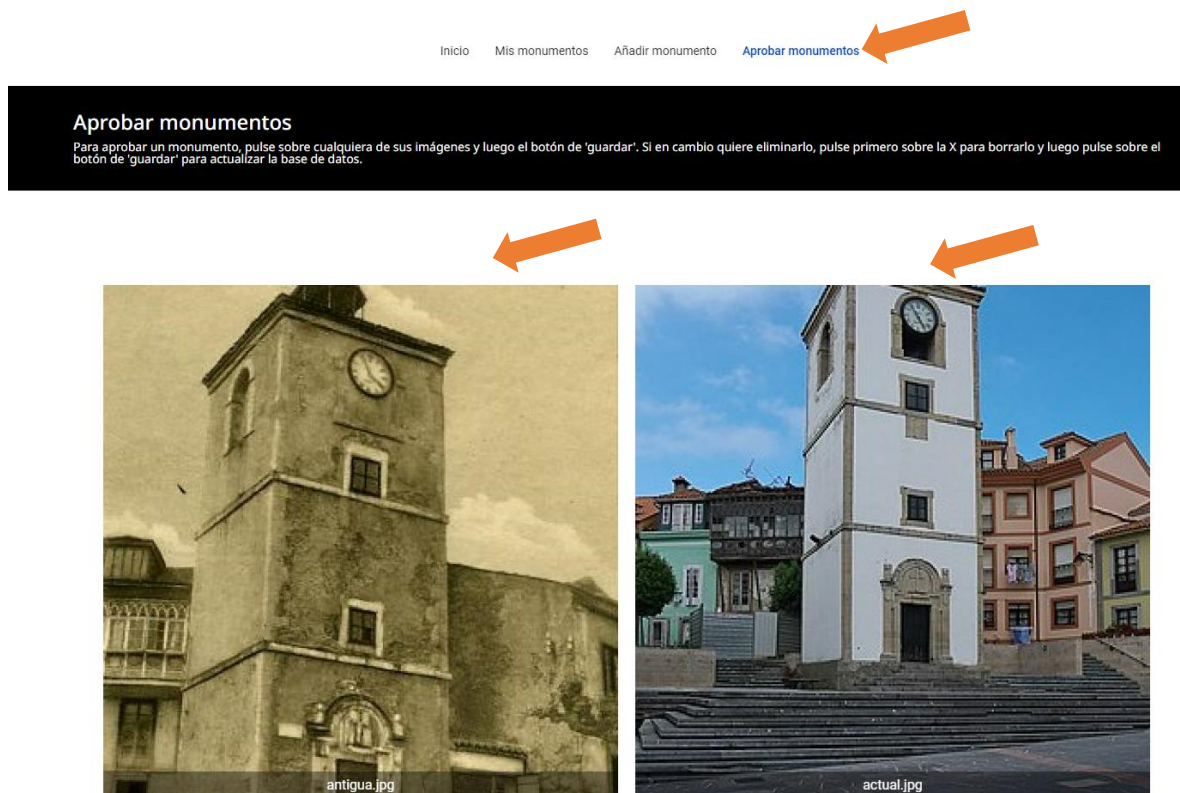


Figure 10.19.- Approve monuments tab

As showed on Fig 10.20, monument image is displayed when an administrator clicks on it. Just click on the indicated button and monument will be added to Vuforia database.



Figure 10.20.- Approved monument

10.2.2 Delete monument

Administrator must follow the same steps of the previous section but, in this case, button pressed will be the one indicated on Fig 10.21.



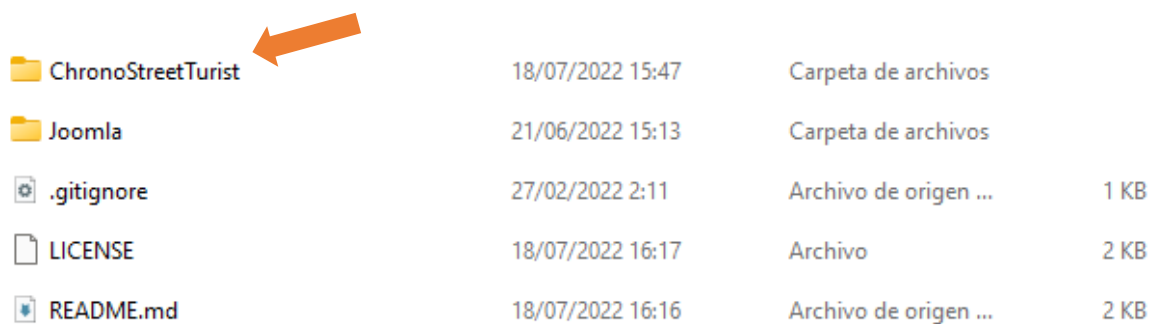
Figure 10.21.- Deleted monument

10.3 INSTALLER MANUAL

Below are the instructions for installer to be able to set up the project.

10.3.1 Import unity project

Unity project is intuitive and easy to import because all dependencies are automatically installed. *Unity* editor version used was *2020.1.14f1* but later versions also support it. Installer must install *Unity Hub* from official *Unity* portal. Then, download the *Unity* project from the repository (Fig. 10.22) and unzip it if necessary.








 ChronoStreetTurist	18/07/2022 15:47	Carpeta de archivos	
 Joomla	21/06/2022 15:13	Carpeta de archivos	
 .gitignore	27/02/2022 2:11	Archivo de origen ...	1 KB
 LICENSE	18/07/2022 16:17	Archivo	2 KB
 README.md	18/07/2022 16:16	Archivo de origen ...	2 KB

Figure 10.22.- Project repository

Finally, open *Unity Hub* and select the project folder when clicking the button indicated on Fig 10.23.

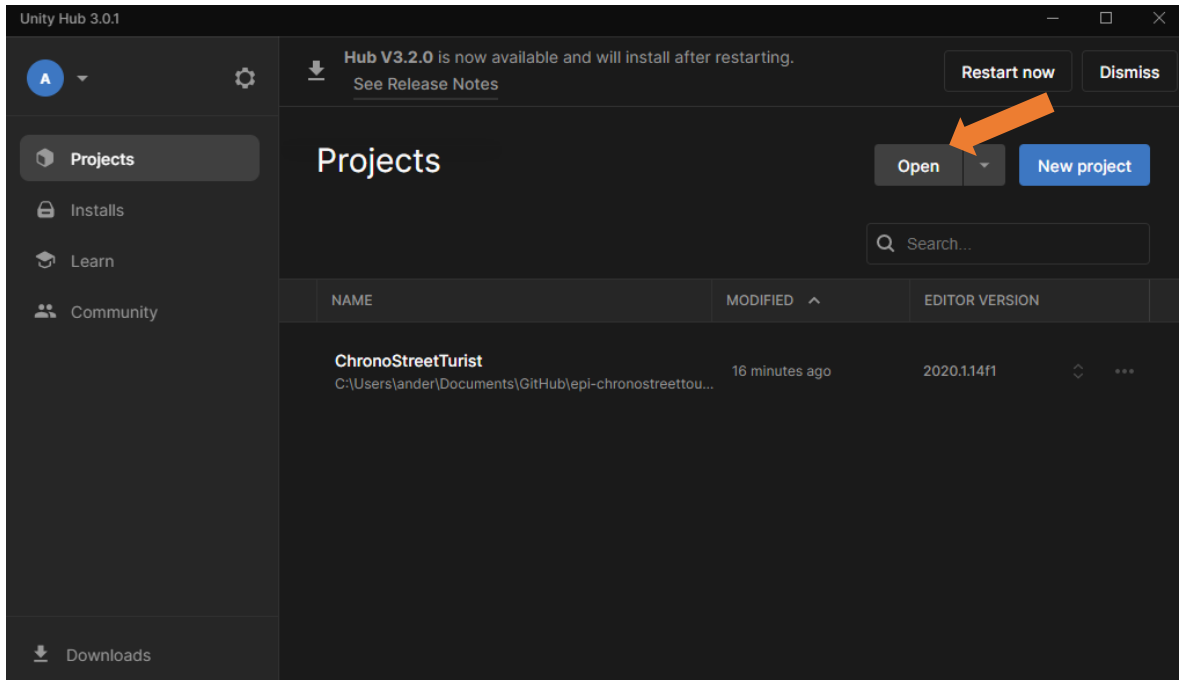


Figure 10.23.- Unity Hub

10.3.2 Set up Vuforia license

First, installer must create a developer account in Vuforia web portal (<https://developer.vuforia.com>).

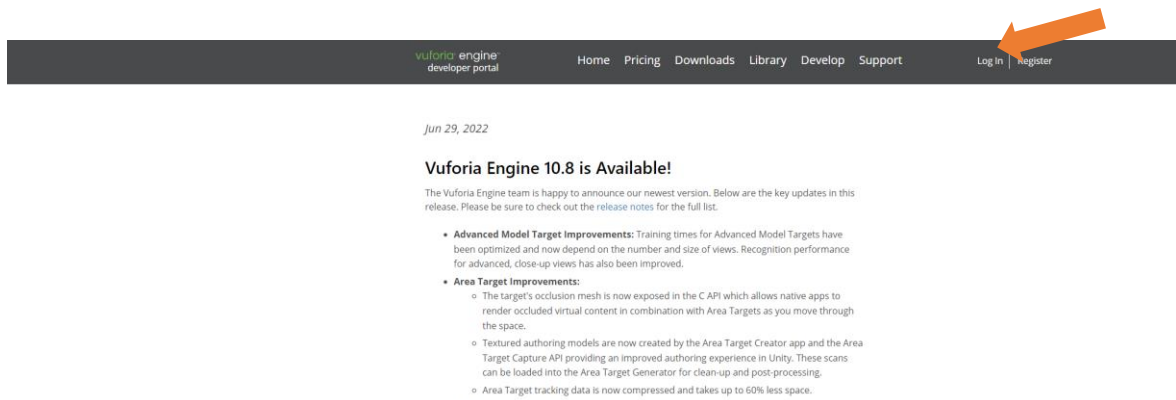
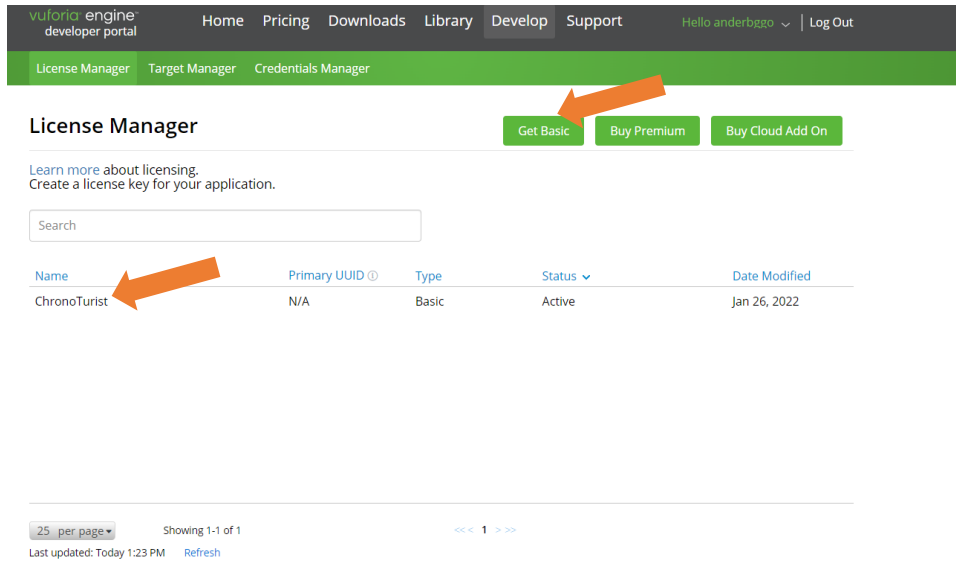


Figure 10.24.- Vuforia developer portal

Next, installer must get a basic license following the steps of the Get Basic button indicated on Fig 10.25. Once license is obtained, click on your license name (second arrow of Fig 10.25) and copy the license name indicated in Fig 10.26.



vuforia engine developer portal

Home Pricing Downloads Library Develop Support Hello anderbggo Log Out

License Manager Target Manager Credentials Manager

License Manager

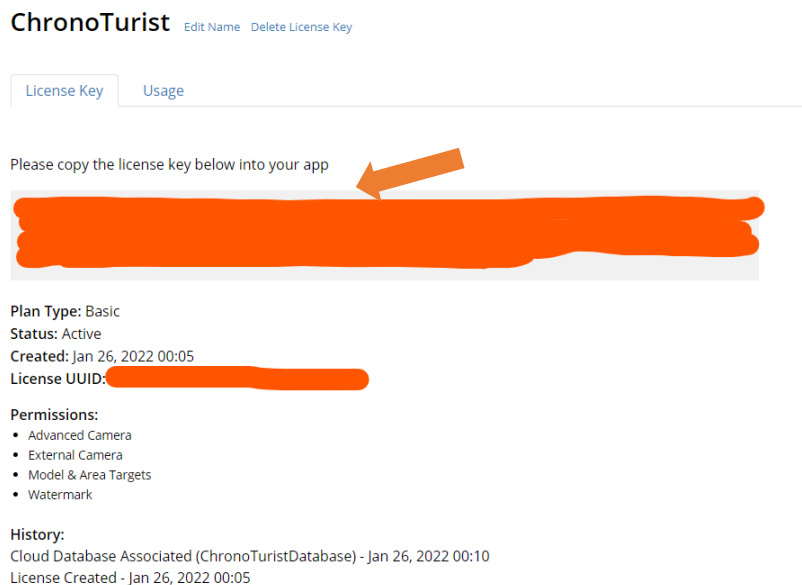
Learn more about licensing.
Create a license key for your application.

Search

Name	Primary UUID	Type	Status	Date Modified
ChronoTurist	N/A	Basic	Active	Jan 26, 2022

25 per page Showing 1-1 of 1 Last updated: Today 1:23 PM Refresh

Figure 10.25.- License manager



ChronoTurist

Edit Name Delete License Key

License Key Usage

Please copy the license key below into your app

Plan Type: Basic
Status: Active
Created: Jan 26, 2022 00:05
License UUID: [redacted]

Permissions:

- Advanced Camera
- External Camera
- Model & Area Targets
- Watermark

History:

Cloud Database Associated (ChronoTuristDatabase) - Jan 26, 2022 00:10
License Created - Jan 26, 2022 00:05

Figure 10.26.- License key

Now installer must go to Unity project and select AR Camera in the left menu as showed in Fig 10.27. Then click on the button showed in Fig 10.28 and paste license in field showed in Fig 10.29.

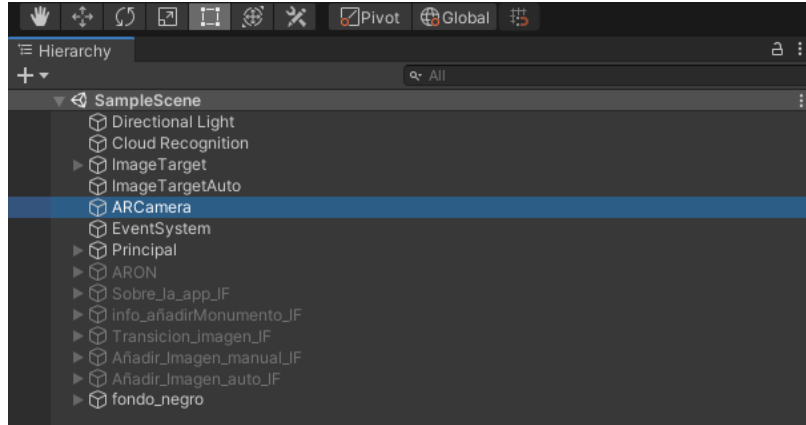


Figure 10.27.- AR Camera element

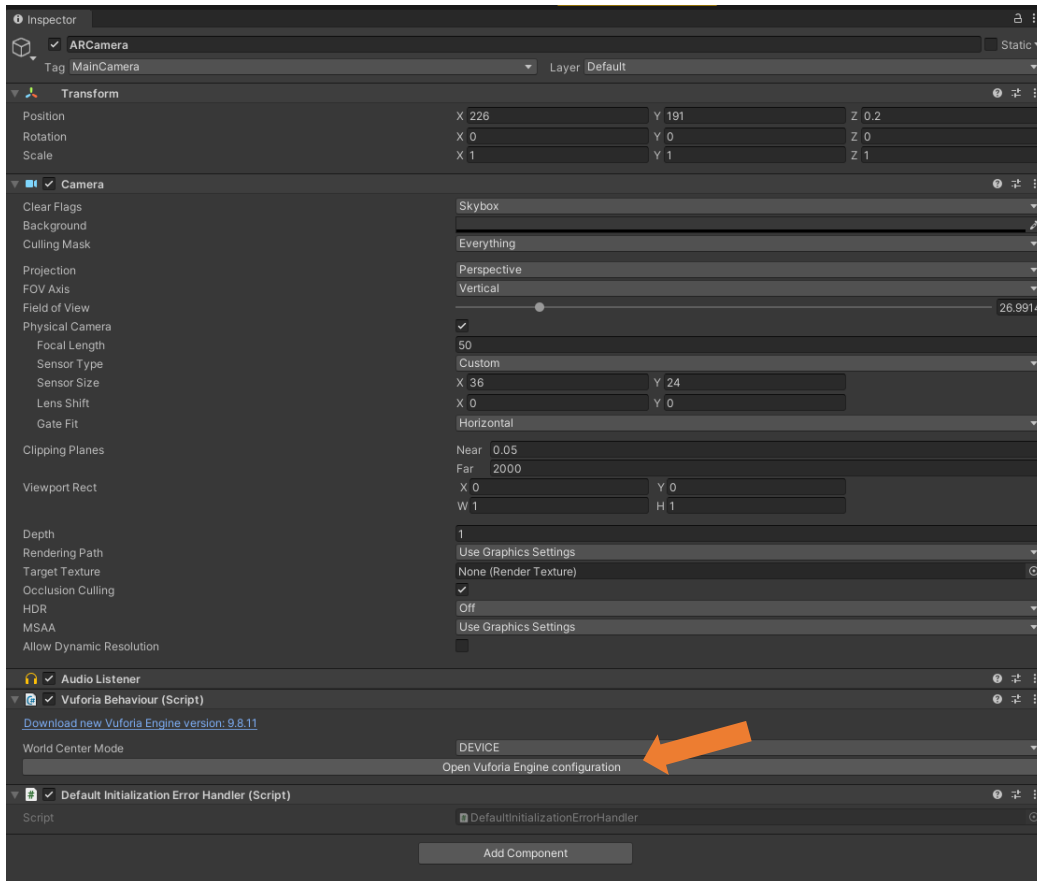


Figure 10.28.- AR Camera properties

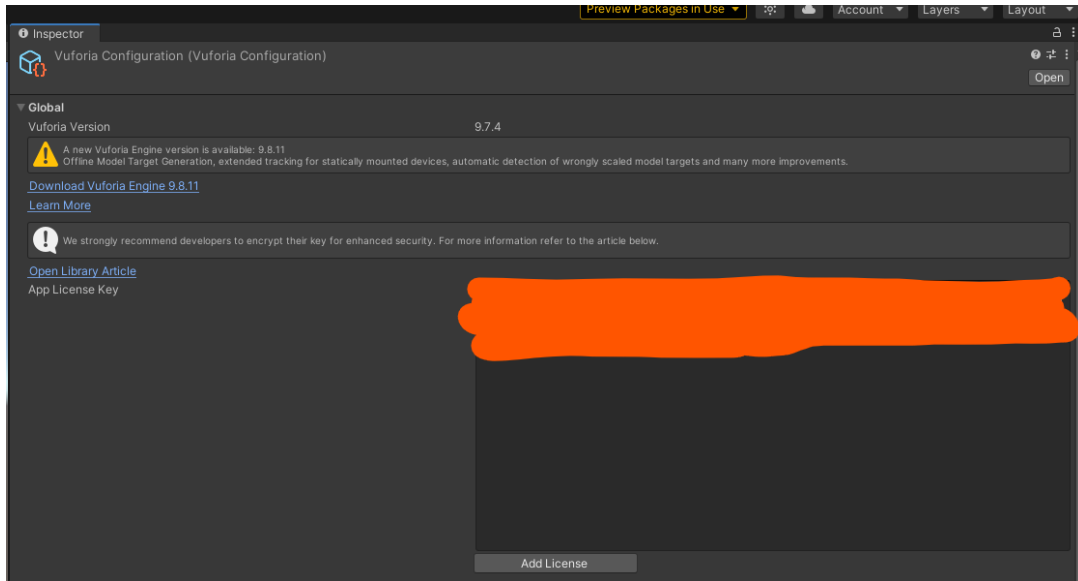


Figure 10.29.- Vuforia engine conf.

10.3.3 Set up Vuforia database

Installer must select Target Manager tab on Vuforia developer portal as showed in Fig 10.30. Then, click on *Add Database*.

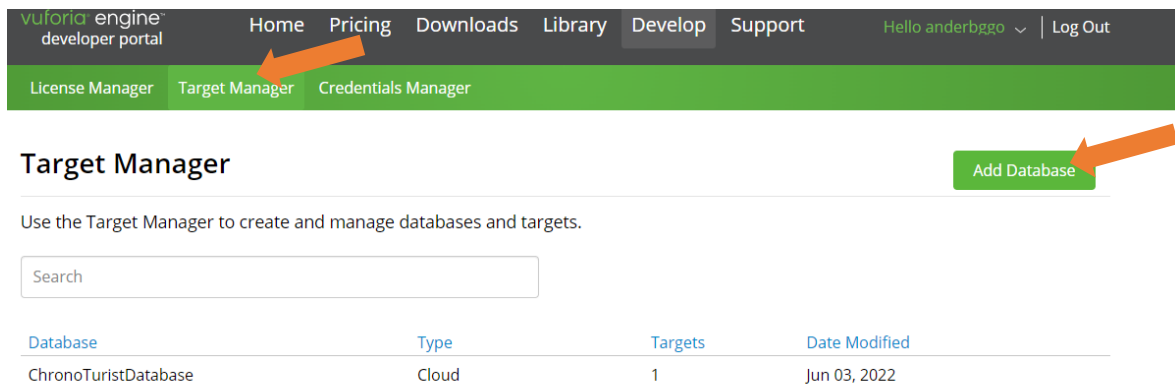


Figure 10.30.- Target manager.

Give a database name, select Cloud in the radio buttons of Fig 10.31 and select the license created on previous section in the dropdown menu.

Create Database

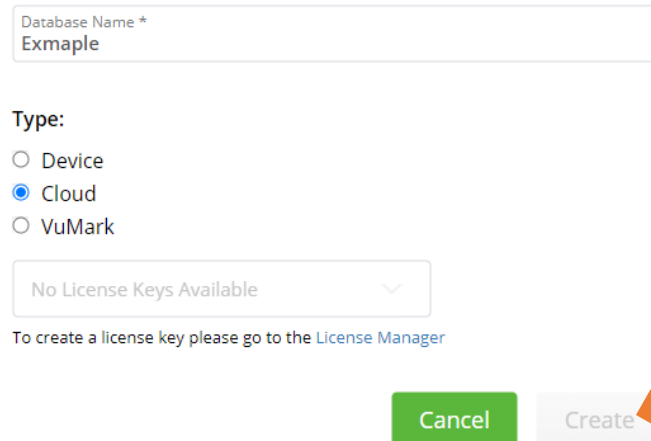


Figure 10.31.- Vuforia database creator.

Once database is created. Enter on it and select Database access keys tab. Copy client and server access keys.

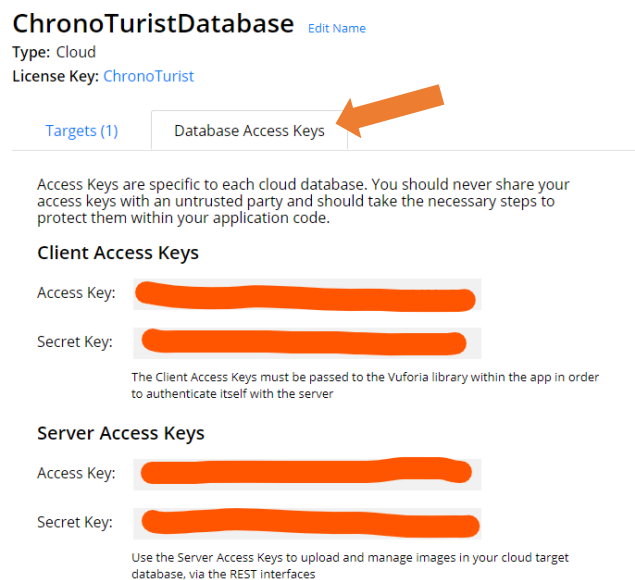


Figure 10.32.- Database access keys.

Next step will be to give app access to the database so it can use the cloud recognition. To achieve this, installer must enter to the *Unity* project and select the element indicated in Fig 10.33.

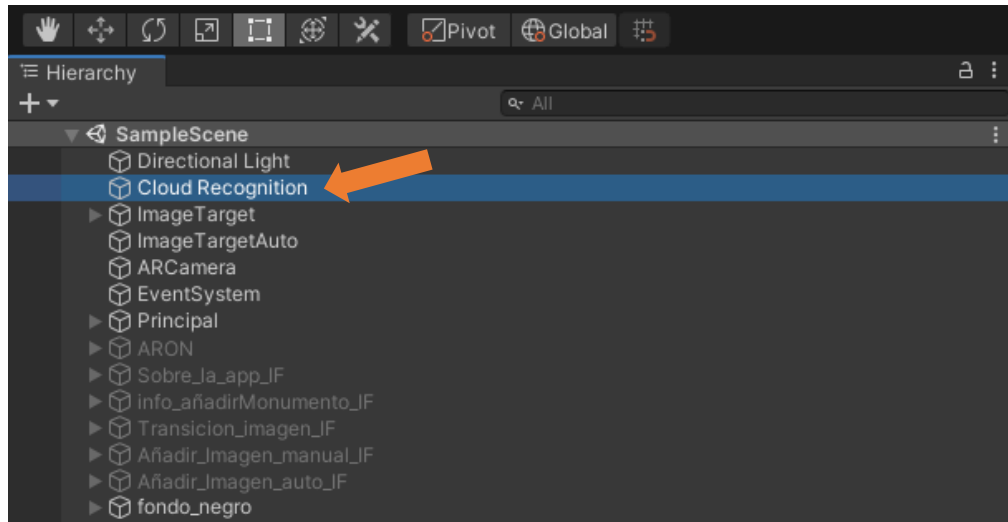


Figure 10.33.- Cloud recognition element.

Then, paste client Access keys on fields indicated in Fig 10.34.

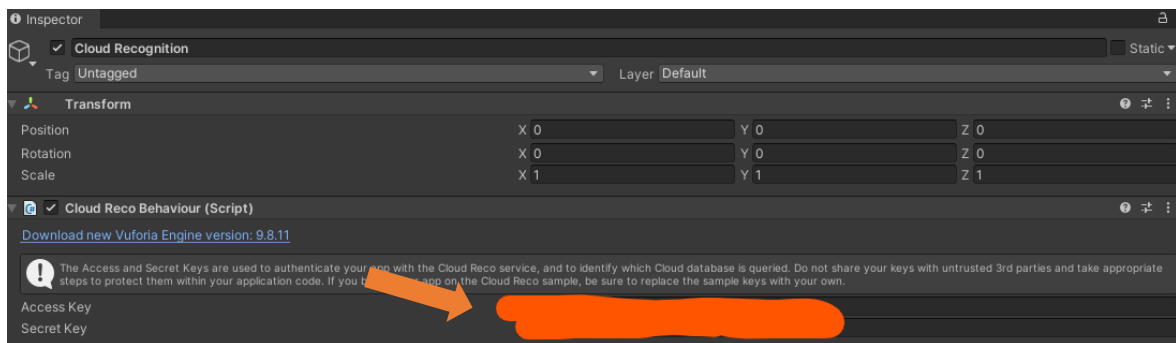


Figure 10.34.- Unity database access keys.

Finally, installer must give access to CMS. For this purpose, go to /htdocs/plugins/user/joomla/joomla.php in the CMS archive files and replace lines 2993 and 2994 with server access keys.



Figure 10.35.- Joomla.php server access keys

11. Technology research

In this section all the technologies used for the project will be analyzed.

11.1 *Vuforia*

Vuforia is a cross-platform Augmented Reality (AR) and Mixed Reality (MR) application development platform, with robust tracking and performance on a variety of hardware. Unity's integration of Vuforia allows to create vision apps and games for Android and IOS.

Vuforia supports many third-party devices (such as AR/MR glasses), and VR devices with back-facing cameras (such as the Gear VR). *Vuforia* can be uses in any device with a camera to test AR/MR games and applications built in *Unity*.



Figure 11.1.- Vuforia logo

11.2 APP DEVELOPMENT PLATFORM

The app has been developed in *Unity* because its free version is very complete, it is one of the most popular platforms and it has a lot of free plugins. It also supports *Vuforia*, which is our main hub.

The programming language used by *Unity* is C#, therefore the development of the app has been carried out entirely in this language.



Figure 11.2.- Unity logo

In addition, a *Unity* plugin called *Visual Studio editor* (developed by *Microsoft*) has been used to be able to debug from VS[\[27\]](#).

11.3 CONTENT MANAGER SYSTEM (CMS)

A content management system is a software application that enables users to create, edit, collaborate on, publish and store digital content. They are typically used for enterprise content management (ECM) and web content management (WCM). To carry out the web service, several options were proposed.

On the one hand, it was contemplated the idea of creating a web service from zero by using programming languages such as *Angular* or frameworks such as *RoR*[\[19\]](#). This idea was quickly discarded due to the lack of knowledge about the technologies and the excessive amount of work for a single person.

On the other hand, most popular CMS were investigated, including *WordPress*, *Joomla* and *Drupal*. *Wordpress* was discarded because its free version does not allow you to install plugins and *Drupal* has very little support and most of it was obsolete. Instead, *Joomla* seemed to have great support and popularity along with lots of free plugins to make the tasks easier. For these reasons, *Joomla* was chosen.



Figure 11.3.- Joomla logo

Joomla is mostly written in PHP so, although this technology is beginning to be obsolete, most of the code written has been in this language. Also, HTML[\[11\]](#) and JS[\[13\]](#) were used.

As it was anticipated before, some *Joomla* plugins were used. All of them with GPL and MIT licenses. They will be explained below.

11.3.1 *ConvertForms*

It is a *Joomla* plugin that allows the administrator to create and modify forms. It has a very complete free version that is the one we have used. In addition, it allows administrators to create *PHP* events to perform more complex tasks such as saving monuments in the database once the form is sent or creating metadata files based on the information entered.

For all these reasons this plugin was chosen. In addition, its developer *Tassos Marinos* provides great support and resolves doubts quickly.

11.3.2 *Sigplus*

Sigplus adds photo and multimedia galleries to a *Joomla* page with a simple syntax. Its free version is enough to cover the requirements of the project. In addition, the code is well organized and commented which allows to reprogram its buttons administrators approve or delete the monuments. Its developer, *Levente Hunyadi*, solve doubts in a matter of hours.

11.4 HOSTING

A hosting is a service through which storage and computing resources are providing to an individual for the accommodation and maintenance of one or more websites and related services.

The hosting must be free, have at least 500 Mb^[15] of storage, allow *Joomla* integration along with its plugins and allow HTTP^[12] and FTP^[9] requests. Due to the large number of requirements that the hosting had to be met, countless different of them have been investigated to see which could be the best for the project. In the end, the chosen hosting was CloudAccess.



Figure 11.4.- CloudAccess logo

The chosen hosting meets all the requirements, but its free version has to be renewed every 30 days.

11.5 REPOSITORY

A git repository hosted by *Gitlab* was used during the project development.

Git is a free and open-source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. It has been used due to its popularity and its great help for applying the *scrum* methodology.

Gitlab instead, is an open-source code repository and collaborative software development platform for large DevOps projects. The reason why *Gitlab* was used against other options is because *HP SCDS* company offered an enterprise license with unlimited storage. In addition, *Gitlab* includes a set of extra functionalities that made it a great choice for this project such as sprint definitions (milestones) or user stories definition (issues).



Figure 11.5.- Gitlab logo

11.6 DATABASE

On the one hand, the project uses a free cloud database from the *Vuforia* portal (target manager) in which approved monuments are stored. This database is used because it is the place in which *Vuforia* service searches for images to do cloud recognition.

In addition to this, our project needs another database to store pending monuments uploaded by users. For this reason, several options were considered.

The first version of the project used *Google Drive* folders as a database (different folder for each monument files). But this method was discarded because, due to complex *Google Drive* authentication, there was no easy way to upload and download images from there (each user should have access to the *Google* account and that is dangerous). Other similar services like *Dropbox* or *Onedrive* had the same problem because they are not intended to be used as a database.

At this point another solution appeared: directly upload the images to the hosting. This was the chosen solution due to its simplicity and effectiveness.

11.7 TEXT EDITORS

During the project development, different text editors have been used. *Visual Studio* was used during the development of the app because it allowed debugging. In addition, *Visual Studio code* was used for convenience when opening a script quickly. Lastly, *Notepad++* was used for file editing in the web service. Again for convenience.



Figures 11.6. and 11.7- Visual Studio and Notepad++ logos

11.8 FTP CLIENT

WinSCP was used for this task. Other options like *Filezilla* would also be valid.



Figure 11.8.- WinSCP logo

11.9 DOCUMENTATION

To do the documentation, we use principally *Microsoft Office* (*Word*, *Excel*, *PowerPoint*). Also, other tools were used as *GanttProject* or *Draw.io*.



Figure 11.9.- Microsoft office

12. Budget

Budget is divided in 3 different categories, whose expenses will later be summed up as the total expected cost of the project. Also, as the service is not for sale, possible benefit has been excluded from this calculation, but should be considered if situation changes. VAT[26] and other taxes are included in each step.

12.1 SOFTWARE

Most of the software used for the development such as *Notepad++* or *VSCode*[28] is distributed as open source. Others like *Unity* or *CloudAcces* were used under its free distribution licenses. Finally, the cost of *Gitlab* cannot be calculated as it is a company license and price paid is unknown to those outside of it. In the following table, non-free software used is shown:

ID	Concept	Quantity (u)	Price per unit (€)	Total (€)
1.1	Microsoft Windows 11 Home License	1	145.00	145.00
1.2	Microsoft Office 2019 License	1	149.00	149.00
			Total	294.00

Table 12.1.- Software budget

With a total software price of 294.00€, it has to be taken into account the real depreciation suffered during the project duration. Estimating a total software lifetime of 60 months, and with this project having a duration of about 6 months, the real software cost in terms of depreciation is:

$$\text{Software cost with depreciation} = 294.00\text{€} \times \frac{6 \text{ months}}{60 \text{ months}} = 29.40\text{€}$$

The software final cost is 29.40€ (TWENTY-NINE EUROS AND FORTY CENTS).

12.2 HARDWARE AND OTHERS

The following table shows the original price of each article used for the development of the project:

ID	Concept	Quantity (u)	Price per unit (€)	Total (€)
2.1	MSI Stealth 15M A11SDK-081FR	1	1,689.56	1,689.56
2.2	Monitor Millenium MD27PRO	1	296.57	296.57
2.3	Keyboard Corsair K95RGB Platinum	1	139.97	139.97
2.4	Headphones Corsair HS60 Pro Surround	1	55.00	55.00
2.5	Mouse DELL MS116-BK	1	10.00	10.00
2.6	Pens, paper and photocopies		15.00	15.00
			Total	2,206.10

Table 12.2.- Hardware budget

As in the previous section case, the amortized cost has to be calculated taking into account the lifespan of the software. Assuming again 60 months of use, and with the project going on for 6 months, the real hardware cost in terms of depreciation is:

$$\text{Hardware cost with depreciation} = 2,206.10\text{€} \times \frac{6 \text{ months}}{60 \text{ months}} = 220.61\text{€}$$

The hardware final cost is 220.61€ (TWO HUNDRED TWENTY EUROS AND SIXTY-ONE CENTS).

12.3 HUMAN RESOURCES

In this section, the time dedicated (measured in hours) of the developer and the tutors is taken into account.

ID	Concept	Quantity (hours)	Price per hour (€)	Total (€)
3.1	Developer's hours	450	25.00	11,250.00
3.2	Tutor's hours	20	30.00	600.00
3.3	Co-tutors' hours	50	30.00	1,500.00
			Total	13,350.00

Table 12.3.- Human resources budget

The total cost of labor hours is 13,350.00€ (THIRTEEN THOUSAND THREE HUNDRED FIFTY EUROS).

12.4 TOTAL

In this section, the final project budget is calculated based on the previous ones.

ID	Concept	Total (€)
4.1	Software budget	29.40
4.2	Hardware budget	220.61
4.3	Human resources	13,350.00
Final price		13,600.01

Table 12.4.- Final budget

The final presented budget for the project amounts to 13,600.01€ (THIRTEEN THOUSAND SIX HUNDRED EUROS AND ONE CENT).

13. Proposed improvements

During the development, interesting ideas and improvements for the project have arisen. Many of these could not be implemented due to time and budget constraints but they could be interesting for a future development.

13.1 *Google Street View*

One of the goals of the project was to investigate how to integrate *Google Street View* with monuments uploaded by the user, so I will use that section to expose the conclusions. The first obstacle was noticing that *Google Maps API*[\[1\]](#) is not free and, although it has a trial version, it requires entering payment data. For this reason, alternatives were investigated.

13.1.1 *Bing StreetSide View*

Bing Streetside view offers a 360-degree view of streets, sidewalks, buildings, and natural elements in a given location (similar to *Google Street View*). It is free, but it only has data of *Madrid, Barcelona, Zaragoza* and *Valencia*. It is clearly a good alternative although it would limit the scope of the project.

13.1.2 *Mapillary*

Mapillary is a service for sharing crowdsourced geotagged photos, developed by remote company *Mapillary AB*. It was launched in 2013 and acquired by *Meta* in 2020. It is one of the few alternative platforms that has street level imagery like *Google Street View*.

It is free, but it does not show the interiors of the cities and has less data than the competition. Anyway, it is an interesting alternative.

13.2 USER LOGIN FROM THE APP

Another of hosting limitation was to be able to make logins from third-party applications. For this reason, when a user uploads a monument from the app, service only checks that username exists, causing a security breach since any user could upload images with the username of another user.

This issue could be solved by changing hosting or using other mechanisms to verify that the username and password of a user are correct. For example, a file could be created in the hosting (which could only be accessed by FTP) that includes a "signature" of each user. Every time one is added or removed; this file would be automatically updated. When the application wants to verify a user, it must check that his signature is in this file.

13.3 INCREASE BUDGET

The project has great scalability so if budget is increased, its use could be extended to more users, a domain could be acquired, more space in the database, better plugins, etc. In addition, another developer could be hired to implement new functionalities.

13.4 USABILITY

Apart from previously mentioned, there are small changes that would greatly increase the usability of the service:

- Appearance: is not one of the strong points.
- Internationalization: At this moment, the only supported language is Spanish. Translations into other languages would be welcome.
- IOS support: IOS users cannot enjoy the application at this time.

14. Conclusions

This project has been one of my biggest challenges of all times. All the requirements specified by the tutors have been met and, personally, I think I have done a great job.

During the project, there have been good moments when I always wanted to go a little further, but also, others in which I have gotten stuck to the point of despair and thought that I would not be able to keep moving forward. Fortunately, I have been able to find alternatives to the problems that arose during the development.

One of the biggest problems was being able to do POST HTML requests from PHP to *Vuforia*. There were no libraries able (and due to the hosting, they could not be installed) and the POST request required complex things like encoding images in base64, generate authentication token, etc. The solution was to prepare the HTTP request manually and, with patience, it end up working perfectly (*core.php* line 2933). Another big problem was finding a way for administrators to approve or delete the monuments. Solution was to reprogram buttons already existent in *sigplus* plugin.

In addition, this Final Degree Project has supposed a great step forward in my professional development since I have acquired new skills in matters unknown before starting. Also, thanks to the degree, my learning ability is much faster and better.

Finally, I want to acknowledge the great success of being co-tutored by the *HP Technology Observatory*. They have helped me more than I thought.

15. Glossary

- API: *Application Programming Interfaces*.
- APK: *Android Application Package*.
- AR: *Augmented Reality*.
- Cloud recognition: Is the online recognition solution for large-scale Augmented Reality projects. Database is stored in the cloud which allows to store more elements.
- CMS: *Content management system*.
- CMS database: Is a relational database in which user monuments are stored until an administrator deletes them.
- DB: *Database*.
- Feature: reference points used in a target to achieve its recognition. The more features, better rating.
- FTP: *File transfer protocol*.
- GPL: *General Public License* is a license type intended to guarantee your freedom to share and change all versions of a program.
- HTML: *Hyper-Text Markup Language*.
- HTTP: *Hyper-Text transfer Protocol*
- JS: *JavaScript*.
- JSON: *JavaScript Object Notation*.
- MB: *MegaByte*.
- MIT: Is another license type that can be integrated into GPL license. It puts only very limited restrictions.
- Monuments: we refer with monument to the set of old image, current image and metadata of the real monument.
- Rating: score given to a target based on the number of features^[8] found. This ranges from 1 to 5 and the higher it is, the easier it will target recognized.
- RoR: *Ruby on Rails*.
- SDK: *Software Development Kit*.
- SR: *System requirement*.
- TLD: *Top-level domain*. Examples: *.es*, *.com*, *.net*, etc.

- UFR: User functional requirement.
- UNFR: User non-functional requirement.
- URL: Universal Resource Locator.
- VAT: Value Added Tax.
- VS: *Visual Studio*.
- VSCo: *Visual Studio Code*.
- Vuforia database: Database in which approved monuments are stored. Mobile app uses it for cloud recognition.

16. Bibliography

1. ChronoStreetTurist website
<https://chronostreetturist.joomla.com/>
Last Checked: 07/14/2022
2. HP Technology Observatory
<https://hpscads.com/observatorio-hp/observatorio-hp-21-22/#ChronoStreetTurist>
Last Checked: 07/14/2022
3. Gitlab repository
<https://gitlab.com/HP-SCDS/Observatorio/2021-2022/chronostreettourist/epi-chronostreettourist/>
Last checked: 07/14/2022
4. Vuforia developer portal
<https://developer.vuforia.com/>
Last Checked: 07/14/2022
5. Diagram.io to make diagrams
<https://app.diagrams.net/>
Last Checked: 07/14/2022
6. GanttProject to make Gantt diagram
<https://www.ganttproject.biz/>
Last Checked: 07/14/2022
7. Official Unity website
<https://unity.com/>
Last Checked: 07/14/2022

8. Unity official forum
<https://forum.unity.com/>
Last Checked: 07/14/2022

9. CloudAccess (Hosting used)
<https://ccp.cloudaccess.net/>
Last Checked: 07/14/2022

10. Drupal official website
<https://www.drupal.org/>
Last Checked: 07/14/2022

11. Wordpress official website
<https://wordpress.com/>
Last Checked: 07/14/2022

12. Joomla community
<https://community.joomla.org/>
Last Checked: 07/14/2022

13. ConvertForms plugin official website
<https://www.tassos.gr/joomla-extensions/convert-forms>
Last Checked: 07/14/2022

14. Sigplus plugin official website
<https://extensions.joomla.org/extension/sigplus/>
Last Checked: 07/14/2022

15. Bing StreetSide View
<https://www.bing.com/api/maps/sdkrelease/mapcontrol/isdk/setviewtostreetside>
Last Checked: 07/14/2022

16. Mappillary python SDK

<https://github.com/mapillary/mapillary-python-sdk>

Last Checked: 07/14/2022

17. Google Street View API

<https://developers.google.com/maps/documentation/streetview/overview>

Last Checked: 07/14/2022

18. Unity-Google Drive repository

<https://github.com/Elringus/UnityGoogleDrive>

Last Checked: 07/14/2022

19. App icon

<https://thenounproject.com/icon/monument-2218141/>

Last Checked: 07/14/2022

20. Other logos

<https://www.flaticon.com/free-icons/info>

<https://www.flaticon.es/iconos-gratis/boton-de-inicio>

Last Checked: 07/14/2022

21. Stack overflow to investigate code solutions

<https://stackoverflow.com/>

Last Checked: 07/14/2022

22. CatonMat to investigate code solutions

<https://catonmat.net/>

Last Checked: 07/14/2022

23. Reqbin to test HTTP requests

<https://reqbin.com/>

Last Checked: 07/14/2022

24. [Github repository about CURL requests](https://gist.github.com/subfuzion/08c5d85437d5d4f00e58)
<https://gist.github.com/subfuzion/08c5d85437d5d4f00e58>
Last Checked: 07/14/2022

25. [C# official documentation](https://docs.microsoft.com/es-es/dotnet/csharp/)
<https://docs.microsoft.com/es-es/dotnet/csharp/>
Last Checked: 07/14/2022

26. [PHP official documentation](https://www.php.net/manual/es/index.php)
<https://www.php.net/manual/es/index.php>
Last Checked: 07/14/2022

27. [HTTP official documentation](https://httpwg.org/specs/)
<https://httpwg.org/specs/>
Last Checked: 07/14/2022

28. [JavaScript documentation](https://developer.mozilla.org/en-US/docs/Web/JavaScript)
<https://developer.mozilla.org/en-US/docs/Web/JavaScript>
Last Checked: 07/14/2022

29. [RoR official website](https://rubyonrails.org/)
<https://rubyonrails.org/>
Last Checked: 07/14/2022

30. [Base64 encoder and decoder](https://www.base64encode.org/)
<https://www.base64encode.org/>
Last Checked: 07/14/2022

31. [Tab Nine to search for code solutions](https://www.tabnine.com/)
<https://www.tabnine.com/>
Last Checked: 07/14/2022

32. Wikipedia to check for information about monuments

[https://es.wikipedia.org/wiki/Torre_del_Rejoj_\(Luanco\)](https://es.wikipedia.org/wiki/Torre_del_Rejoj_(Luanco))

[https://es.wikipedia.org/wiki/Iglesia_de_Santa_Mar%C3%ADa_\(Luanco\)](https://es.wikipedia.org/wiki/Iglesia_de_Santa_Mar%C3%ADa_(Luanco))

[https://es.wikipedia.org/wiki/Catedral_de_Notre_Dame_\(Par%C3%ADs\)](https://es.wikipedia.org/wiki/Catedral_de_Notre_Dame_(Par%C3%ADs))

Last Checked: 07/14/2022

33. Imgur to search for images

<https://imgur.com/>

Last Checked: 07/14/2022

34. Aurasma

<http://raenelaula.blogspot.com/p/aurasma.html>

<https://universoabierto.org/2016/09/02/aurasma-app-de-realidad-aumentada-para-insertar-una-historia-en-un-libro-en-el-espacio-fisico/>

Last Checked: 07/14/2022

35. Augmented Reality for visitors of cultural heritage sites. In Proc. of Int. Conf. on Cultural and Scientific Aspects of Experimental Media Spaces, 2001

Pages 89–93

Didier Stricker, John Karigiannis, Ioannis T Christou, Tim Gleue and Nikos Ioannidis.

36. Augmented Reality. End of grade project, 2021

Universidad de Burgos

José Antonio Cerrato Casillas