



UNIVERSIDAD DE BURGOS  
ESCUELA POLITÉCNICA SUPERIOR  
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería  
Informática**

**Diseño de un sistema de  
identificación de personas  
Documentación Técnica**



Presentado por Víctor de Castro Hurtado  
en Universidad de Burgos — 10 de junio  
de 2018

Tutor: César Represa



---

# Índice general

---

<b>Índice general</b>	<b>I</b>
<b>Índice de figuras</b>	<b>III</b>
<b>Índice de tablas</b>	<b>v</b>
<b>Apéndice A Plan de Proyecto Software</b>	<b>1</b>
A.1. Introducción . . . . .	1
A.2. Planificación temporal . . . . .	2
A.3. Estudio de viabilidad . . . . .	22
<b>Apéndice B Especificación de Requisitos</b>	<b>23</b>
B.1. Introducción . . . . .	23
B.2. Objetivos generales . . . . .	24
B.3. Catalogo de requisitos . . . . .	25
B.4. Especificación de requisitos . . . . .	26
<b>Apéndice C Especificación de diseño</b>	<b>27</b>
C.1. Introducción . . . . .	27
C.2. Diseño de datos . . . . .	27
C.3. Diseño procedimental . . . . .	27
C.4. Diseño arquitectónico . . . . .	27
<b>Apéndice D Documentación técnica de programación</b>	<b>29</b>
D.1. Introducción . . . . .	29
D.2. Estructura de directorios . . . . .	29
D.3. Manual del programador . . . . .	29

D.4. Compilación, instalación y ejecución del proyecto . . . . .	29
D.5. Pruebas del sistema . . . . .	29
<b>Apéndice E Documentación de usuario</b>	<b>31</b>
E.1. Introducción . . . . .	31
E.2. Requisitos de usuarios . . . . .	31
E.3. Instalación . . . . .	31
E.4. Manual del usuario . . . . .	31
<b>Bibliografía</b>	<b>37</b>

---

## Índice de figuras

---

A.1. Primer commit. . . . .	3
A.2. Commit con el que finaliza la parte del planteamiento del proyecto. . . . .	3
A.3. Commit con los recursos instalados en el proyecto. . . . .	8
A.4. Commit con el que finaliza la instalación de recursos. . . . .	8
A.5. Commit con las primeras imágenes de prueba. . . . .	9
A.6. Commit con el que finaliza la fase de abrir, capturar, guardar y modificar una imagen. . . . .	9
A.7. Commit que solucionaba el problema de recursos no compatibles con las nuevas versiones de python. . . . .	9
A.8. Commit con la implementación de la captura de la cámara. . . . .	10
A.9. Commit con la localización del rostro en una imagen implementada. . . . .	10
A.10. Commit con la normalización de la iluminación de la imagen. . . . .	10
A.11. Reconocimiento facial utilizando el método LBPH. . . . .	11
A.12. Commit con cambios en la implementación de los métodos que permiten el reconocimiento facial (parte I). . . . .	11
A.13. Commit con cambios en la implementación de los métodos que permiten el reconocimiento facial (parte II). . . . .	11
A.14. Commit con cambios en la implementación de los métodos que permiten el reconocimiento facial (parte III). . . . .	12
A.15. Commit con cambios en la implementación de los métodos que permiten el reconocimiento facial (parte I). . . . .	13
A.16. Commit con la implementación de ventana de la nueva interfaz. . . . .	13
A.17. Commit con el menú final de cara al usuario final. . . . .	14
A.18. Commit con la implementación de la barra de progreso. . . . .	14
A.19. Commit con el 'popup' y la información extra. . . . .	14
A.20. Commit solucionando el problema de las ventanas 'popup'. . . . .	14
A.21. Commit con el ejecutable base y su mejora. . . . .	15

A.22.Commit con la primera versión de la memoria en LaTeX. . . . .	16
A.23.Commit con una de las actualizaciones del fichero README.md. . . . .	16
A.24.Grafo con el número de commits por semana. . . . .	17
A.25.Flujograma inicial de primer nivel. . . . .	18
A.26.Flujograma inicial de 2º nivel. . . . .	18
A.27.Flujograma inicial de tercer nivel: Capturar imagen. . . . .	19
A.28.Flujograma inicial de tercer nivel: Localizar rostro. . . . .	19
A.29.Flujograma inicial de tercer nivel: Extraer características. . . . .	20
A.30.Flujograma inicial de tercer nivel: Extraer patrón. . . . .	20
A.31.Flujograma inicial de tercer nivel: Entrenar red. . . . .	21

---

# Índice de tablas

---





## Apéndice A

---

# Plan de Proyecto Software

---

### A.1. Introducción

Aunque nos hemos basado en un método de **desarrollo en cascada** [3], hemos incluido algunos elementos de la **metodología scrum** [4] como las reuniones mensuales.

En cuanto al desarrollo en cascada, hemos tomado la base de este método, ya que teníamos unos **requisitos iniciales**, que eran los objetivos primarios **B.4** que teníamos que cumplir, a partir de los cuales se hizo un primer **diseño del proyecto**, tras lo cual se pasó al **desarrollo e implementación**.

En cuanto a la metodología scrum, incluimos reuniones con el tutor. Las primeras para ver que estructura tenía el proyecto y que posibles cambios iniciales se podrían realizar, y el resto de ellas para ver cómo iba avanzando el proyecto, lo que necesitaba ser mejorado o cambiado, o asignar nuevas tareas hasta la próxima reunión (en nuestro caso la mayoría fueron semanales o quincenales, en vez de mensuales).

## A.2. Planificación temporal

Las fases que se han seguido a la hora de desarrollar el proyecto han sido [2]:

### ■ Definir el proyecto

Al principio se tuvo que escoger el tipo de proyecto que se iba a realizar. En nuestro caso un proyecto medio, ya que no se tenían ni el tiempo ni los recursos necesarios para que fuera un proyecto demasiado grande, y tampoco podía ser un proyecto demasiado pequeño dado el objetivo del mismo.

También se tuvo que definir la idea principal, que en nuestro caso iba a ser reconocer gente.

### ■ Identificar información, recursos y requisitos

A continuación, se definió qué tipo de información iba a tratar: íbamos a trabajar con imágenes en sus diferentes formatos, y con redes neuronales, las cuales íbamos a entrenar para que reconocieran a la gente almacenada en nuestra base de datos.

Los recursos de los que íbamos a disponer eran:

1. La imagen que obtuviéramos con la cámara.
2. Otras imágenes de gente a la que necesitáramos reconocer (en nuestro caso gente desaparecida o en busca y captura). Dichas imágenes se obtendrían de una base de datos oficial de la policía/gobierno.
3. Información básica sobre cada una de las personas que tuviéramos almacenadas en la base de datos de nuestro programa.

En cuanto a los requisitos que necesitábamos cumplir, se establecieron los siguientes (se entrará en más detalle en la sección *B\_Requisitos B*):

1. Obtener imágenes en tiempo real.
2. Entrenar una red neuronal con machine-learning o derivados (deep learning, computer vision, etc).

3. Tener una base de datos con imágenes que usaríamos para entrenar dicha red.
4. Realizar una comparación satisfactoria de la persona identificada en la imagen en tiempo real.
5. Mostrar resultados e información en una interfaz al usuario.

Estas primeras fases del proyecto (identificación de recursos y requisitos, planteamiento del proyecto, definición y diseño del proyecto) duró aproximadamente **dos semanas**, desde la primera de Febrero que se creó el proyecto en Github [A.1](#), hasta el 18 del mismo mes, que se pasó a la fase de desarrollo al empezar a instalar el material necesario [A.2](#).



Figura A.1: Primer commit.

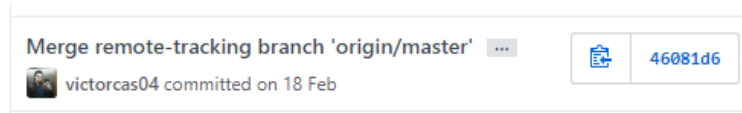


Figura A.2: Commit con el que finaliza la parte del planteamiento del proyecto.

#### ■ Fase de planificación

Cuando se empezó a desarrollar el proyecto, se sabía cual era el objetivo general del proyecto, aunque no el camino exacto que iba a seguir, de manera que se optó por seguir una serie de hitos principales, enumerados en el apartado anterior, y a partir de esos hitos ir creando otros más pequeños y asequibles a la hora del desarrollo.

No se pretendía ir completando los hitos uno por uno y que no se pudiera empezar a desarrollar uno hasta que se acabara el anterior, de manera que el progreso se ha ido realizando sobre todos ellos de

manera más o menos equivalente.

Los hitos grandes no tenían fecha prevista de ser completados, sino que se completarían cuando no quedaran tareas pendientes relacionadas con dicho hito, de manera que era complicado planificar el desarrollo para que coincidiera con unos plazos concretos, así que se planificaron el resto de tareas más pequeñas, completando varias para cada reunión.

Dichas reuniones se llevaban a cabo en función del trabajo planteado de una a otra, así que no eran todas las semanas, sino que se adaptaban un poco al trabajo pendiente, aunque si que es cierto que se han tenido reuniones cada (como máximo) dos semanas, para llevar un seguimiento más o menos regular del desarrollo y que no se quede el mismo estancado en el mismo punto demasiado tiempo.

En las herramientas que se han barajado utilizar respecto a la planificación, se barajó utilizar *Trello* [1], aunque se descartó ya que, sin otros miembros que aporten contenido, no tenía sentido tener una lista de tareas 'to do' o 'doing' para una sola persona.

También se pensó en utilizar *Zenhub* [5] para la planificación de hitos, tareas y sprints, aunque se descartó la idea al no tener unos plazos fijos de entrega para cada una de ellas.

#### ■ Fase de investigación

A la vez que se realizaba la planificación de las tareas semanales, se iba realizando un proceso de investigación e información, tanto de conceptos y mecánicas como de seguimiento de tutoriales, documentación y diferentes referencias sobre los temas tratados en el proyecto (para más información consultar el apartado *3\_Conceptos\_teoricos* de la memoria principal).

#### ■ Desarrollo

En este apartado hemos ido realizando el desarrollo del propio proyecto, tanto la estructura de la fase de planificación como el propio código. Para ello se ha seguido un orden lógico de desarrollo, centrándonos primero en la funcionalidad básica de tomar una imagen de la cámara y compararla con las de la base de datos, para pasar más tarde a

ampliar el número de muestras de nuestra red para mejorar el ratio de acierto de las predicciones, crear un interfaz gráfico para mostrar los resultados y que al usuario le resulte más sencilla su interpretación y crear un ejecutable para evitar tener que ejecutar nuestro código completo en un editor.

Estas últimas modificaciones se toman para facilidad del usuario, intentando automatizar todo lo posible el proceso, aunque es cierto que se necesitan ciertos datos por parte del usuario, además de permitirle cierta libertad (por ejemplo, podríamos entrenar la red a cada ejecución, incluso si nuestras muestras no han variado y la red está correctamente entrenada, pero supondría una pérdida de tiempo y recursos en muchos casos, de manera que se opta por darle opción al usuario).

En la tabla A.2 se pueden observar tanto los milestones principales del desarrollo del proyecto como las issues o tareas asociadas a cada uno.

|c|c|c| Distribución del desarrollo del proyecto según Milestones e Issues.

<b>Milestones</b>	<b>Issues</b>	<b>Type of Issue</b>
Tabla A.0 – <i>Continued from previous page</i>		
<b>Milestones</b>	<b>Issues</b>	<b>Type of Issue</b>
<i>Continued on next page</i>		
1.- Install and Configure	Install environment	Develop
	Create class diagram	Develop
	Basic readme	Develop
2.- Image manipulation	Open image from file	Develop
	Save images	Develop
	Tensorflow Tutorials	Develop
	Meeting	Meeting
3.- Basic image recognition	Import OpenCV	Develop
	Capture image from camera	Develop
	Normalize illumination	Enhancement
	Create basic parameters	Develop
	Facial recognition (I)	Develop
	Meeting	Meeting
4.- Apply computer vision	Improve camera recording	Enhancement
	Change face location display	Enhancement
	Facial recognition (II)	Develop
	Meeting	Meeting
	Acquire camera	Enhancement
5.- Create database	Create local storage	Develop
	Get more images	Enhancement
6.- Interface	Basic interface	Develop
	Dynamic window	Enhancement
	Facial recognition (III)	Enhancement
	Camera view	Enhancement
	Result images	Develop
	Compare bar/percentage	Develop
	Meeting	Meeting
	Result information	Develop
	Executable	Develop
7.- Report	Basic report	Develop
	Meeting	Meeting

Clean and order code   Enhancement  
Update readme   Enhancement  
Final report   Enhancement  
Video   Develop

El tiempo que nos ha llevado esta fase ha sido el restante desde finales de Febrero hasta finales de Mayo (casi **tres meses**), siendo la parte más importante del proyecto, a continuación se detalla el tiempo requerido en cada uno de los apartados (*milestones* en Github) en los que hemos dividido el proyecto:

- Instalación y configuración

En este apartado se han incluido pequeñas tareas como pueden ser: instalar el entorno, crear diagrama de clases, crear algunos ficheros básicos del proyecto, etc. En esta fase se instalaron todos los requisitos técnicos, librerías y dependencias.

El tiempo medio empleado en esta fase fue de **una semana**, como se puede ver en el primer [A.3](#) y último commit de esta fase [A.4](#).

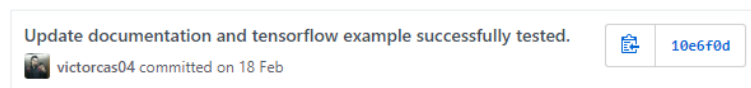


Figura A.3: Commit con los recursos instalados en el proyecto.

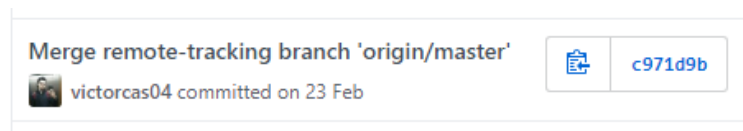


Figura A.4: Commit con el que finaliza la instalación de recursos.

- Manipulación, captura y guardado de imágenes

En este apartado se hacen diversos commits de tareas en las que teníamos que implementar una forma de abrir y guardar imágenes, así como completar algunos tutoriales on-line sobre el funcionamiento de aplicaciones de machine-learning.

El tiempo medio empleado en esta fase fue de **casi un mes**, desde finales de Febrero [A.5](#) hasta mediados de Marzo [A.6](#).



Además, en esta fase se encontró un problema derivado de una de las librerías (*OpenCV*), el cual hacía que dicha librería no fuera compatible con nuestra versión de python (*Python 3.6* en un principio), por lo que se cambió la configuración del proyecto para trabajar con *python 2.7* A.7.

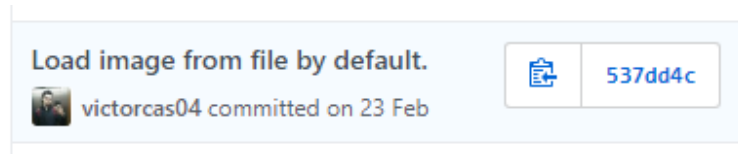


Figura A.5: Commit con las primeras imágenes de prueba.

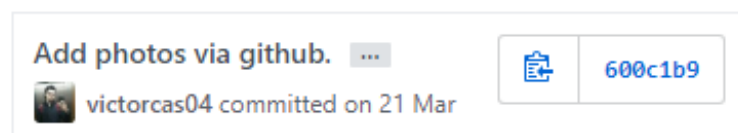


Figura A.6: Commit con el que finaliza la fase de abrir, capturar, guardar y modificar una imagen.

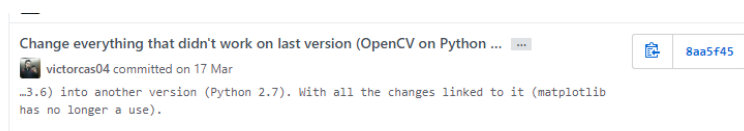


Figura A.7: Commit que solucionaba el problema de recursos no compatibles con las nuevas versiones de python.

- Reconocimiento de imágenes: localización del rostro y extracción de características

Antes de empezar con esta fase, al final de la fase anterior se dejaron los tutoriales de TensorFlow aparte, ya que suponían una inversión enorme de tiempo, y se optó por una opción más sencilla: *OpenCV* A.7.

En este apartado se desarrolla la captura por pantalla de un frame A.8. A continuación, se implementa la primera parte del reconocimiento facial, donde se localizan los rostros de la gente en dicha imagen A.9, previamente normalizada A.10, además de extraer las características de cada uno de esos rostros (en nuestro caso hemos limitado el número de caras que puede reconocer en cada imagen a 1 para evitar falsos positivos).

Además, se implementó una primera versión de reconocimiento facial a partir de la técnica *LBPH* A.11, explicada en la memoria principal, en la sección *3\_conceptos\_teoricos*.

El tiempo medio empleado en esta fase fue de **casi un mes**, desde mediados de Marzo hasta mediados de Abril.

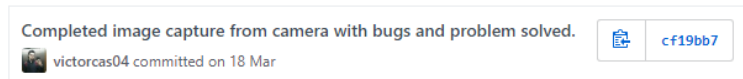


Figura A.8: Commit con la implementación de la captura de la cámara.

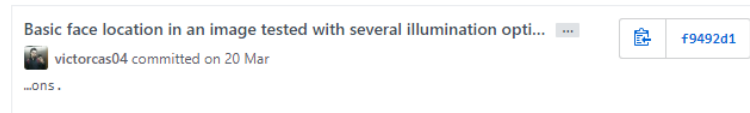


Figura A.9: Commit con la localización del rostro en una imagen implementada.

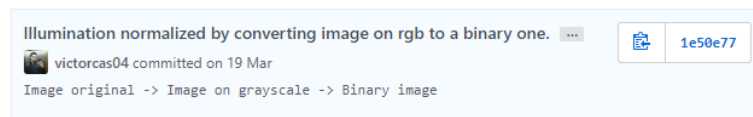


Figura A.10: Commit con la normalización de la iluminación de la imagen.

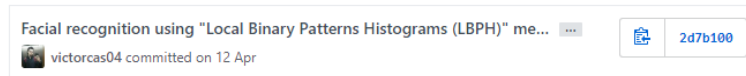


Figura A.11: Reconocimiento facial utilizando el método LBPH..

- Reconocimiento de imágenes: mejorar resultados de predicciones con computer vision

En esta parte se mejora el porcentaje de reconocimiento de una persona (aproximadamente, de un 40 50 % a un 60 70 %) A.12, A.13 restringiendo la zona de obtención de características del rostro y cambiando parámetros tanto de iluminación como de la predicción de la red A.14.

El tiempo medio empleado en esta fase fue de **un mes**, desde mediados de Abril hasta mediados de Mayo.

---

**Auto-recognize people.** Develop  
#27 by victorcas04 was closed on 12 Apr

---

Figura A.12: Commit con cambios en la implementación de los métodos que permiten el reconocimiento facial (parte I).

---

**Improve facial recognition.** enhancement  
#22 by victorcas04 was closed on 23 Mar

---

Figura A.13: Commit con cambios en la implementación de los métodos que permiten el reconocimiento facial (parte II).

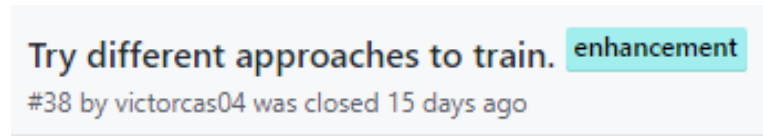


Figura A.14: Commit con cambios en la implementación de los métodos que permiten el reconocimiento facial (parte III).

- Desarrollar una interfaz

Esta es la parte que va a ver el usuario: la interfaz visual, de manera que se han empleado **algo más de dos meses**, desde principios de Marzo hasta mediados de Mayo.

Una de las primeras tareas es la de ajusta el tamaño de las imágenes de salida para hacer dicha salida más estable a la hora de realizar comparativas o ejecuciones sucesivas [A.15](#). También se modifica el grosor del rectángulo que rodea el rostro de manera automática, de manera que sea proporcional al tamaño de la imagen, y se hace una **ventana de tamaño variable** en función del tamaño de la pantalla en medidas relativas, para que dicha interfaz se adapte el dispositivo en que se ejecute [A.16](#).

Se decide crear un pequeño pero intuitivo menú para el usuario. Las funcionalidades descritas en el menú ya estaban funcionales de etapas anteriores, pero se ha decidido incluir dicho menú para facilidad de uso por parte del usuario [A.17](#).

En la ventana donde se muestran los resultados, se ha añadido también en esta etapa una barra de progreso, indicando en tanto por ciento (%) la probabilidad de que sea la persona correcta identificada [A.18](#).

Como un pequeño extra, se ha incluido algo de información de cada una de las personas en la base de datos (nombre, edad, etc), de manera que si se pulsa el botón de *More information*, nos creará una ventana emergente (*popup*) con dicha información [A.19](#).

Durante la implementación de esta tarea, nos encontramos con un problema, y es que no podíamos abrir dos ventanas de la misma

interfaz al mismo tiempo ya que daba fallos de compatibilidad (por motivos de seguridad, para evitar que unas ventanas creen otras de forma recursiva), pero al final se consiguió solucionar este pequeño contratiempo añadiendo algunos parámetros en la creación de este 'popup' para evitar que la ventana principal siguiera ejecutándose A.20.

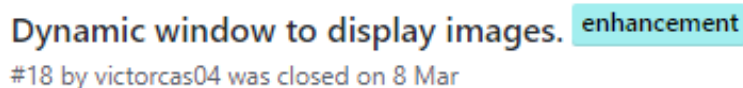
Para terminar con el entorno de usuario, se ha creado un pequeño script ejecutable (fichero *.bat*) que ejecuta el proyecto desde la consola de comandos, mostrando alguna información por dicha consola (más por curiosidad que por utilidad, al usuario medio lo que haga internamente el programa no le importa, pero para este proyecto se ha decidido incluirlo para propósitos académicos), aunque la información importante se mostrará en la interfaz explicada anteriormente A.21.

---



**Image size display.** enhancement  
#21 by victorcas04 was closed on 24 Apr

Figura A.15: Commit con cambios en la implementación de los métodos que permiten el reconocimiento facial (parte I).



**Dynamic window to display images.** enhancement  
#18 by victorcas04 was closed on 8 Mar

Figura A.16: Commit con la implementación de ventana de la nueva interfaz.

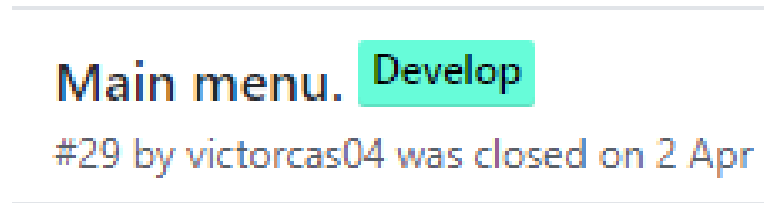


Figura A.17: Commit con el menú final de cara al usuario final.

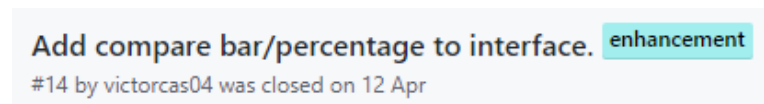


Figura A.18: Commit con la implementación de la barra de progreso.

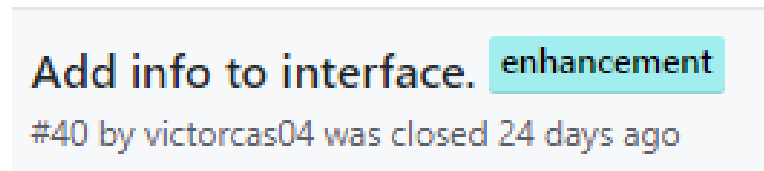


Figura A.19: Commit con el 'popup' y la información extra.



Figura A.20: Commit solucionando el problema de las ventanas 'popup'.

**Improve executable.** enhancement

#39 by victorcas04 was closed 8 days ago

---

**Executable.** enhancement

#37 by victorcas04 was closed 16 days ago

---

Figura A.21: Commit con el ejecutable base y su mejora.

- Memoria y anexos

Esta parte consiste en escribir la memoria con todos los anexos, desarrollando cada punto y dando explicación a algunas preguntas interesantes [A.22](#).

También se ha actualizado el fichero *Readme.md* que sirve como carta de presentación de nuestro proyecto en *Github* [A.23](#).

Esta tarea se lleva desarrollando desde principios de Mayo, y hasta la fecha (principios de Junio) se sigue desarrollando.

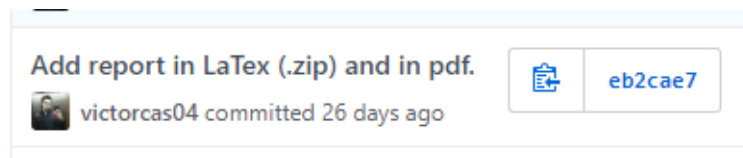


Figura A.22: Commit con la primera versión de la memoria en LaTeX.

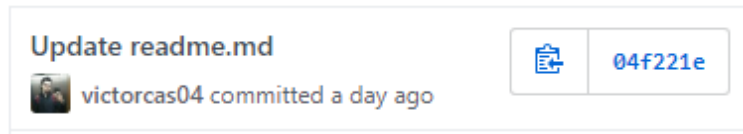


Figura A.23: Commit con una de las actualizaciones del fichero README.md.

### ■ Revisión y control continuo

Aunque esta parte no es una fase la cual haya que planificar como las demás, si que se han ido teniendo reuniones (semanales o bisemanales) mencionadas en la planificación, además de haber ido llevando un control de errores por cuenta propia, que se iban resolviendo inmediatamente si eran urgentes, o después de realizar las tareas asignadas para esa semana si no afectaban al uso general del programa.

Con el proyecto en *Github*, ha sido sencillo seguir el progreso y desarrollo del mismo, sabiendo en todo momento qué tareas estábamos realizando, cuales nos quedaban por realizar, cuales eran los objetivos generales, etc.



Se puede ver un gráfico con el histórico de commits en Github en la imagen [A.24](#).

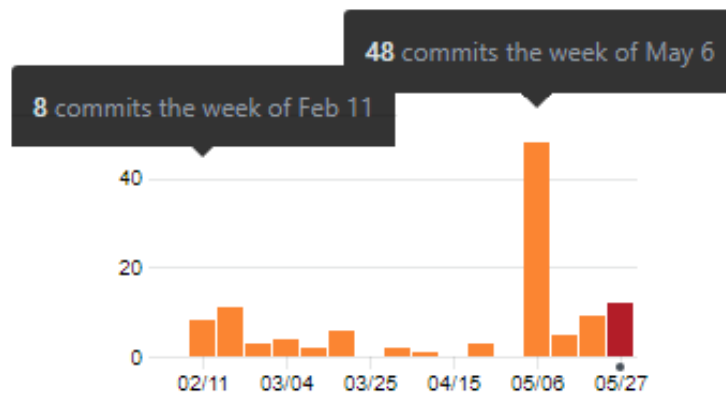


Figura A.24: Grafo con el número de commits por semana.

## Flujogramas y diagramas de clases iniciales

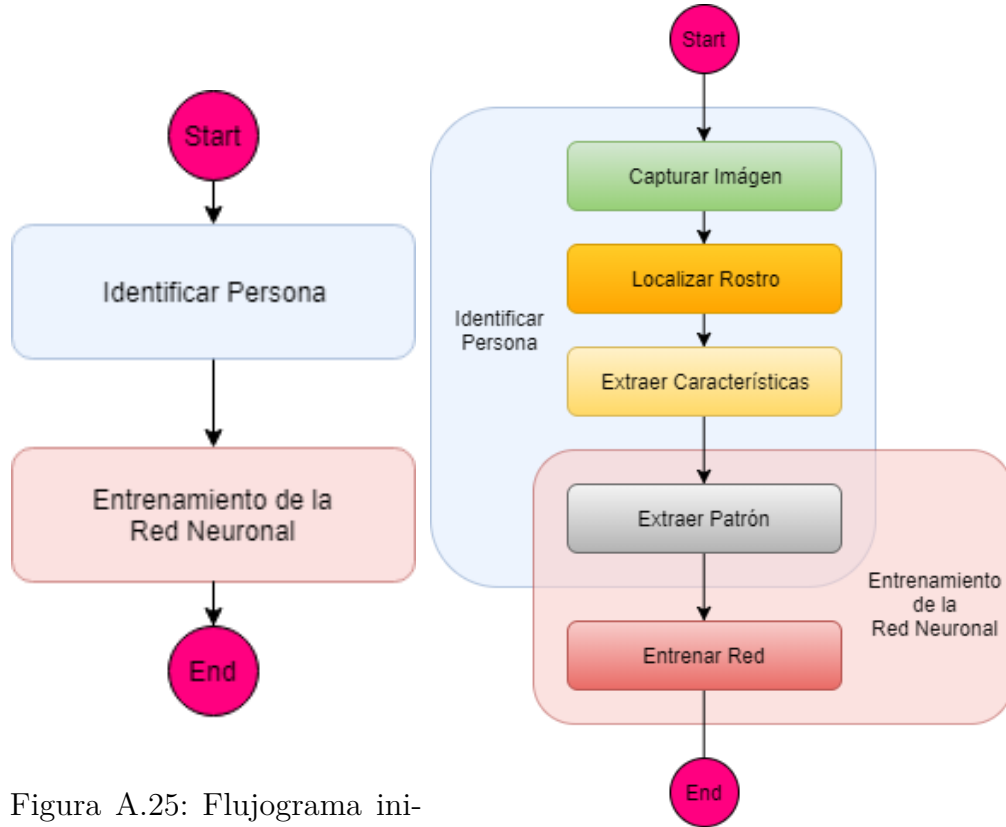


Figura A.25: Flujograma inicial de primer nivel.

Figura A.26: Flujograma inicial de 2º nivel.

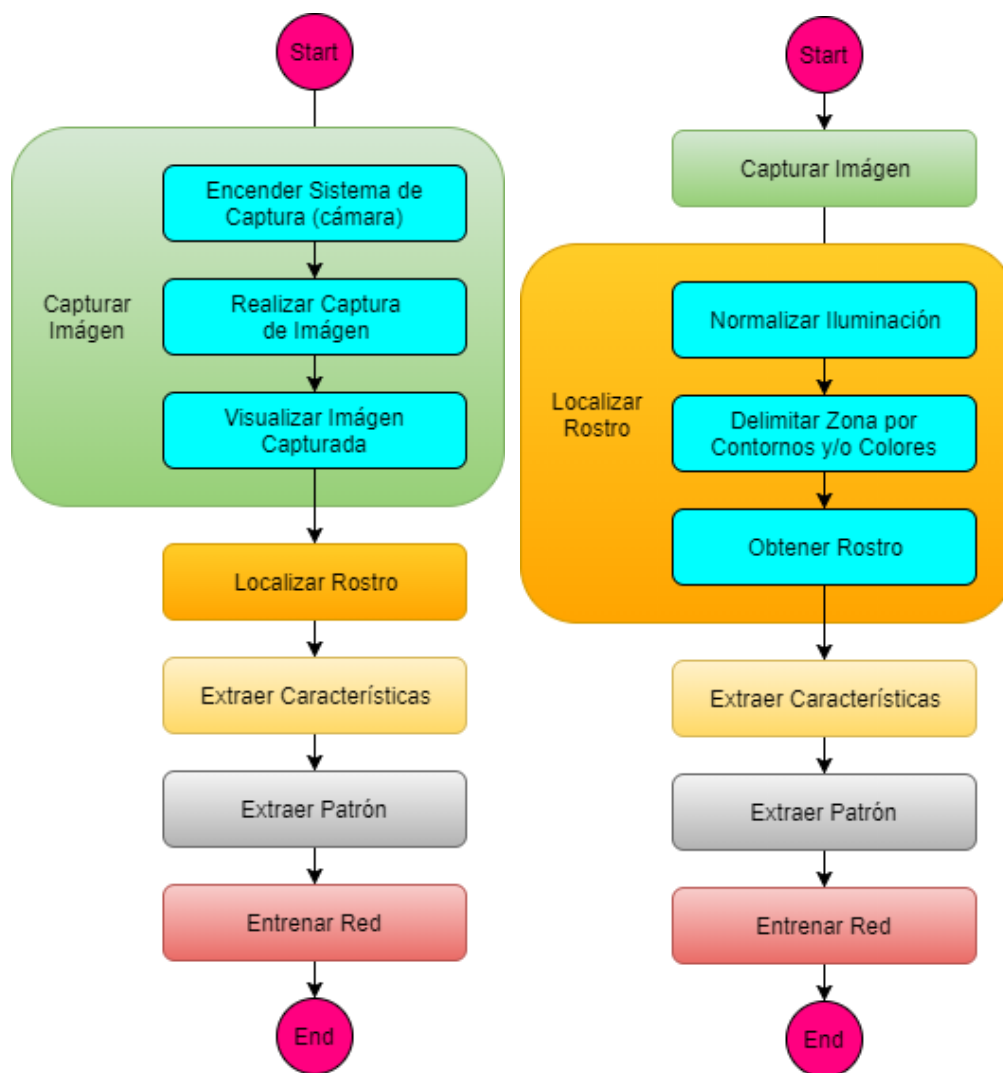


Figura A.27: Flujograma inicial de tercer nivel: Capturar imagen.

Figura A.28: Flujograma inicial de tercer nivel: Localizar rostro.

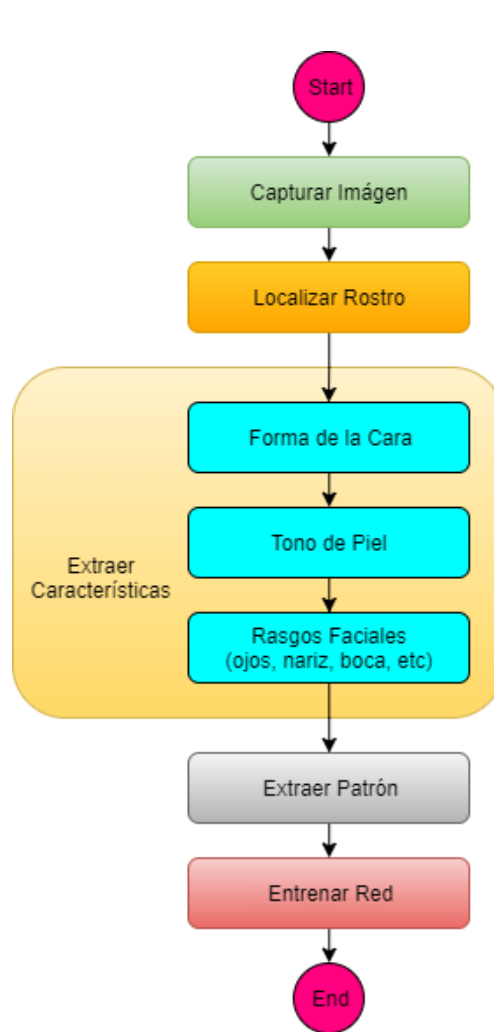


Figura A.29: Flujograma inicial de tercer nivel: Extraer características.

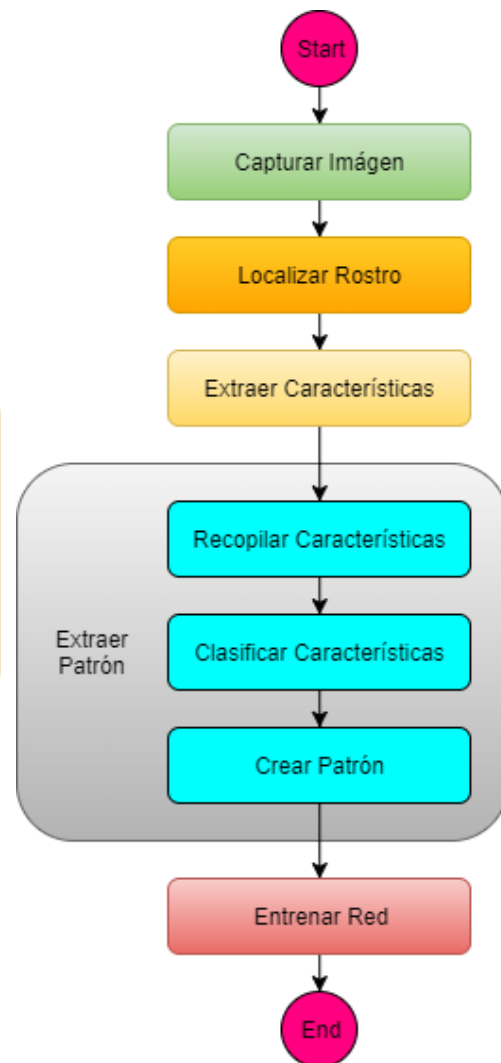


Figura A.30: Flujograma inicial de tercer nivel: Extraer patrón.

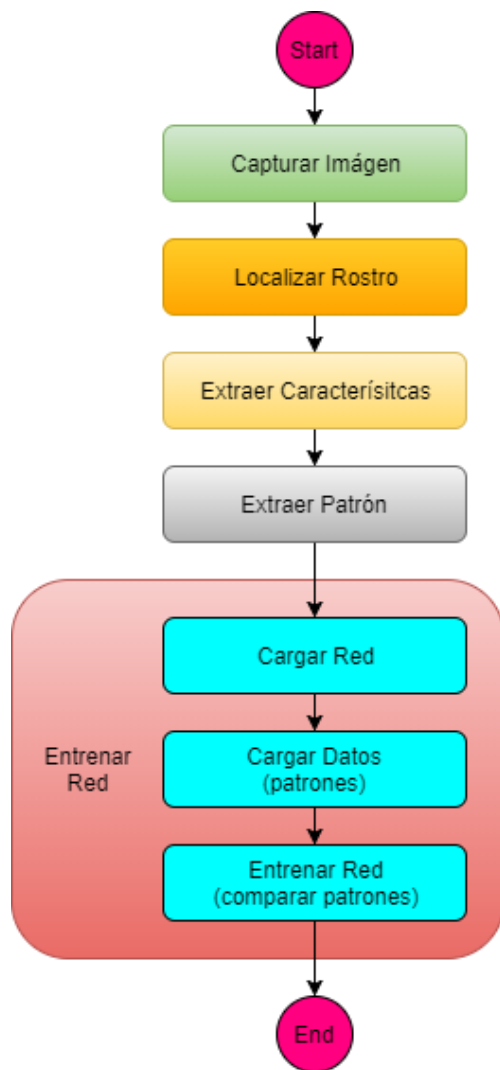


Figura A.31: Flujograma inicial de tercer nivel: Entrenar red.

### A.3. Estudio de viabilidad

¿Es viable?

Esta pregunta es un poco ambigua, y vamos a entrar más en detalle en el anexo *A\_Plan\_proyecto*, pero la respuesta breve es **SI** en el ámbito tecnológico y **DEPENDE** en el ámbito moral, legal y jurídico.

Es viable tecnológicamente si se tiene un *sistema distribuido* [?] en el cual se tenga un servidor (que sea capaz de realizar procesamiento en paralelo [?]) donde se realicen todas las operaciones de comparación, predicción, muestreo de resultados, etc., y varios clientes, concretamente uno por cada cámara (o zona de cámaras), que se encargarán de obtener las imágenes de la calle y enviárselas al servidor junto con información de qué cámara ha realizado esa captura, en qué zona está, a qué hora, etc. Como podemos observar, ya se tiene una estructura similar hoy en día, de modo que se podría aprovechar ese sistema, ahorrando costes.

En cuanto al apartado legal, es una cuestión complicada, ya que, depende de países, unos tienen unas normas y leyes de privacidad más o menos estrictas. En el caso de España, ya se controla el acceso a determinados lugares públicos como aeropuertos, de manera que sería extender ese control y mejorarlo.

Una cuestión que mucha gente no está dispuesta a dejar pasar es el riesgo moral: ¿merece la pena la pérdida de privacidad por un poco más de seguridad? Desde luego, es una cuestión muy importante, pero es demasiado compleja como para resolverlo aquí, ya que cada uno tendrá su opinión y tampoco es el objetivo que persigue este proyecto.

**Viabilidad económica**

**Viabilidad legal**

**Viabilidad moral**

## *Apéndice B*

---

# **Especificación de Requisitos**

---

### **B.1. Introducción**

## **B.2. Objetivos generales**



## **B.3. Catalogo de requisitos**

**B.4. Especificación de requisitos**

## *Apéndice C*

---

# **Especificación de diseño**

---

- C.1. Introducción
- C.2. Diseño de datos
- C.3. Diseño procedimental
- C.4. Diseño arquitectónico



## *Apéndice D*

---

# **Documentación técnica de programación**

---

- D.1. Introducción
- D.2. Estructura de directorios
- D.3. Manual del programador
- D.4. Compilación, instalación y ejecución del proyecto
- D.5. Pruebas del sistema



## Apéndice *E*

---

# Documentación de usuario

---

### E.1. Introducción

### E.2. Requisitos de usuarios

### E.3. Instalación

### E.4. Manual del usuario

Para ejecutar el programa, simplemente hay que descargarse el proyecto (podemos hacer un clone a una carpeta local desde *Github* o simplemente descargarlo como *.zip*) y dar doble click sobre el fichero llamado *run.bat* que encontramos en la carpeta principal.

Esto nos abrirá una consola de comandos donde nos irán saliendo mensajes informativos sobre el progreso del programa, aunque la parte principal creará ventanas para mostrar la interfaz y que la información aparezca de forma más visual y entendible para el usuario medio.

Las principales fases por las que pasa nuestro programa son:

#### Añadir nuevas imágenes

- Nos preguntará si queremos añadir una imagen nueva a la base de datos.

- En caso de responder [Y] (yes), se inicializará la cámara por defecto del equipo.

*NOTA:* La cámara por defecto es la webcam en caso de tener integrada, externa en caso contrario. En caso de tener ambas cámaras instaladas, se utilizará la integrada.

- Podremos tomar una imagen pulsando [C] o salir sin tomar ninguna pulsando [Q].

## Entrenar red

- A continuación, si hemos guardado una imagen nueva en la base de datos en el apartado anterior E.4, pasaremos directamente al punto en que se entrena la red, en caso contrario nos preguntará si queremos entrenarla o no.
- En caso de decir que NO, se utilizarán los recursos almacenados del último entrenamiento.
- En caso de decir que SI, se crearán las imágenes en el formato adecuado y se entrenará la red, utilizando los recursos recién creados.

**IMPORTANTE:** si se añaden imágenes previas a la ejecución del programa a la base de datos (antes del primer apartado E.4), entrenar la red para evitar problemas sobre identificaciones erróneas.

## Carga de recursos e inicialización

- Se cargan los recursos necesarios: *trainerData.yml* y *haarcascade\_frontalface\_default.xml*.
- A continuación, se le pregunta al usuario cómo desea obtener la imagen que va a contrastar con la base de datos: desde fichero E.4 o desde la cámara E.4.

## Obtener imagen desde fichero

- Si se selecciona esta opción, se abrirá una pequeña interfaz en la que el usuario puede buscar la imagen que quiera en el sistema.



- Una vez que se encuentra la imagen, basta con dar doble click sobre ella o seleccionarla y pulsar aceptar.

*NOTA:* Se puede filtrar el tipo de ficheros que se pueden ver para facilitar la búsqueda (restringido a *.png* y *.jpg*).

## Obtener captura

- Al seleccionar esta opción, si hay más de una cámara instalada nos dejará elegir qué cámara utilizar. En caso contrario utilizará la cámara por defecto.

*NOTA:* La cámara por defecto es la webcam en caso de tener integrada, externa en caso contrario.

- Con la cámara seleccionada se inicializan los parámetros para la captura de imágenes por defecto (fps, tamaño de la imagen, etc).

*NOTA:* en caso de no poder inicializar la cámara con estos parámetros, mostrará un mensaje avisándonos y pasará a la fase E.4 con una imagen por defecto.

- Con la cámara inicializada correctamente, aparecerá una ventana con la imagen en tiempo-real capturada con la cámara.
- Cuando el usuario tiene el control sobre la cámara, puede:
  - Pulsar [Q] para salir sin tomar ninguna imagen.
  - Pulsar [I] para mostrar información sobre esa imagen y el menú de la cámara.
  - Pulsar [P] para pausar la captura de imágenes. Mientras se esté en modo pausa el programa se queda en *stand-by*, y no se toman acciones hasta que se reanuda la ejecución.
  - Pulsar [ESPACIO] mientras está en modo pausa para reanudar la ejecución.
  - Pulsar [C] para tomar una captura y seguir ejecutando el programa.

*NOTA:* no se puede tomar una captura a no ser que haya exactamente una persona delante de la cámara.

Se toma esta decisión para evitar problemas a la hora tanto de entrenar la red como de mostrar coincidencias con la base de datos.

*NOTA:* si se pulsa [Q] se pasará a la fase 5 con una imagen por defecto.

## Comparar imágenes

- Con la imagen obtenida de la fase anterior E.4, se analiza dicha imagen y se comparan los resultados con los obtenidos de entrenar las imágenes almacenadas en la base de datos (fichero *trainerData.yml* de la fase de carga de recursos E.4).
- Se obtiene una **ID**, una **ETIQUETA** (nombre de la imagen) y un **PORCENTAJE** de coincidencia entre ambas imágenes (la mayor coincidencia de la base de datos).
- Se carga la imagen de la base de datos con la etiqueta del apartado anterior y la información relacionada con la id obtenida del fichero *info.txt*.

## Mostrar resultado

- Se muestra una ventana con ambas imágenes, la obtenida de la captura/fichero y la de máxima coincidencia de la base de datos.

*NOTA:* las imágenes se recortan para mostrar sólo el rostro, de esta manera se evitan los tamaños de imágenes excesivamente grandes/pequeños.

- Se muestra además el porcentaje de coincidencia que hayan tenido, y una barra de progreso que indica dicho porcentaje junto con el nombre de la imagen de la base de datos.

*NOTA:* el nombre que se muestra sobre la imagen de la derecha es el de la persona a la que corresponda dicha fotografía (extraído del fichero *info.txt*), mientras que el nombre que se muestra sobre la barra de progreso es el correspondiente al nombre del fichero de dicha foto. Se muestran ambos nombres en caso de que el usuario necesite acceder a esa foto manualmente.

- Además, se crea un botón *More information*, que al pulsarle crea una pequeña ventana extra con información sobre la imagen resultado.

Esta información se puede editar en el fichero *info.txt* mencionado en la fase anterior [E.4](#).

**IMPORTANTE:** si se añade una imagen a la base de datos, es necesario añadir una línea en el fichero *info.txt* con la información de la persona que acabamos de introducir en la base de datos. Esta línea deberá ir en la posición exacta en la que se encuentre la imagen en relación al resto. Por ejemplo, si añadimos una imagen a la que nombramos *aaa.jpg* será la primera en la base de datos, por lo que deberemos añadir una línea al principio del fichero *info.txt*. Si añadimos una imagen que se llame *zzz.jpg* a la base de datos será la última, por lo que necesitaremos añadir una línea al final del fichero *info.txt*. En caso de no realizar este paso, cuando se muestren los resultados, se hará con la información de otra persona.

*NOTA:* esta parte de introducir información sobre cada persona en un fichero en la posición exacta es muy tedioso y peligroso, por ello se está estudiando la posibilidad de crear una base de datos real para almacenar esta información, de manera que cada imagen tenga vinculada su información de manera automática.



---

## Bibliografía

---

- [1] Atlassian. Trello. <https://trello.com>.
- [2] sinnaps. Metodología de un proyecto. <https://www.sinnaps.com/blog-gestion-proyectos/metodologia-de-un-proyecto>, 2018.
- [3] Wikipedia. Desarrollo en cascada — wikipedia, la enciclopedia libre, 2018. [Internet; descargado 30-mayo-2018].
- [4] Wikipedia. Scrum (desarrollo de software) — wikipedia, la enciclopedia libre, 2018. [Internet; descargado 30-mayo-2018].
- [5] ZenHub. Zenhub. <https://www.zenhub.com>.