universidad de león

Departamento de Matemáticas

MÁSTER UNIVERSITARIO EN INVESTIGACIÓN EN CIBERSEGURIDAD

Trabajo de Fin de Máster

DESARROLLO DE UN SISTEMA DE IDENTIFICACIÓN MEDIANTE TÉCNICAS DE RECONOCIMIENTO FACIAL

Autor: Borja Simancas Ruesgas

Tutor: Adriana Suárez Corona

Cotutor: David Arroyo Guardeño

(Septiembre, 2019)

UNIVERSIDAD DE LEÓN
Departamento de Matemáticas
MÁSTER UNIVERSITARIO EN INVESTIGACIÓN EN CIBERSEGURIDAD
Trabajo de Fin de Máster

ALUMNO: Borja Simancas Ruesgas

TUTOR: Adriana Suárez Corona y David Arroyo Guardeño

TÍTULO: Desarrollo de un sistema de identificación mediante técnicas de reconocimiento facial

CONVOCATORIA: Septiembre, 2019

RESUMEN:
Los sistemas de identificación han cobrado cada vez más importancia en los últimos años, convirtiéndose en una tarea básica en el día a día de la sociedad. Dichos sistemas han evolucionado a lo largo del tiempo ofreciendo alternativas de identificación, como el reconocimiento facial, que antes no eran posibles. El uso de Deep Learning para abordar la problemática de la identificación facial es muy eficiente y supera, en muchos casos, la precisión humana. Además de las ventajas que ofrecen las técnicas Deep Learning, el registro de las identificaciones realizadas en una arquitectura basada en Blockchain, permite que la información permanezca inmutable a lo largo del tiempo y tenga una trazabilidad completa en una infraestructura descentralizada y sin la necesidad de contar con entidades externas que proporcionen confiabilidad. El sistema de identificación desarrollado en este proyecto tiene su base en la aplicación, por un lado, de técnicas Deep Learning sobre reconocimiento facial y, por otro, de una arquitectura Blockchain. La integración de ambas tecnologías demuestra cómo se puede abordar el desarrollo de un sistema sencillo que permite su aplicación en diferentes casos de uso.

Palabras clave: Deep Learning, identificación, facial, Blockchain, Hyperledger Fabric, trazabilidad

Firma del alumno:	**VºBº Tutor:**

Índice general

Índice de figuras IV

Índice de tablas VI

Glosario de términos VII

Introducción 1

1. Revisión del estado del arte 5

1.1. Sistemas de identificación y técnicas de reconocimiento 5
1.2. Técnicas tradicionales de Machine Learning 14
1.3. Deep Learning 19
1.4. Técnicas de reconocimiento facial basadas en Deep Learning 25
1.5. Conjuntos de datos para el entrenamiento y validación de técnicas de reconocimiento facial basadas en DL 27
1.6. Trazabilidad e integridad de la información en sistemas de identificación facial 28
1.7. Blockchain 31

2. Gestión de proyecto software 40

2.1. Alcance del proyecto 40
2.1.1. Definición del proyecto 40
2.1.2. Estimación de tareas y recursos 41
2.1.3. Presupuesto 42
2.2. Plan de trabajo 44
2.2.1. Identificación de tareas 44
2.2.2. Estimación de tareas 45
2.2.3. Planificación de tareas 45
2.3. Gestión de recursos 46
2.3.1. Especificación de recursos 46

2.3.2. Asignación de recursos 46
2.4. Gestión de riesgos 46
2.4.1. Identificación de riesgos 46
2.4.2. Análisis de riesgos 47

3. Desarrollo de la solución 49
3.1. Descripción de la solución adoptada 49
3.1.1. Diseño y arquitectura del sistema 50
3.1.2. Flujograma del proceso 53
3.1.3. Limitaciones técnicas 56
3.2. Tecnologías de desarrollo 57
3.3. Modelo de reconocimiento facial 58
3.4. Implementación de la cadena de bloques 67
3.5. Implementación de la solución global 80

4. Evaluación y pruebas de la solución 86
4.1. Proceso de evaluación 86
4.1.1. Forma de evaluación 86
4.1.2. Casos de prueba y análisis de resultados 87

5. Líneas futuras 97

6. Conclusiones 99

Índice de figuras

1.1. Datos de Naciones Unidas sobre la identidad global 6
1.2. Topología de los sistemas de identificación 7
1.3. Reconocimiento facial 10
1.4. Reconocimiento huella 10
1.5. Reconocimiento iris 11
1.6. Reconocimiento de retina 11
1.7. Reconocimiento basado en geometría de mano 11
1.8. Reconocimiento del paso 12
1.9. Reconocimiento de voz 12
1.10. Pulsación de teclas 13
1.11. Reconocimiento de firma 13
1.12. Matriz de confusión 14
1.13. SVM 16
1.14. PCA 17
1.15. Red neuronal 20
1.16. Estructura Deep Learning 21
1.17. Autoencoder 22
1.18. Deep Boltzmann Machine 22
1.19. Red neuronal recurrente 23
1.20. Estructura red neuronal convolucional 24
1.21. Ejemplo capas red neuronal convolucional 24
1.22. AlexNet 25
1.23. VGGNet 25
1.24. GoogleNet 26
1.25. ResNet 26
1.26. SENet 27
1.27. Arquitectura centralizada 30
1.28. Arquitectura descentralizada 30

1.29. Arquitectura distribuida 31
1.30. Estructura de la cadena de bloques 32
1.31. Estructura de un bloque 33
1.32. Porcentaje de uso de los modelos de Blockchain 36

2.1. Esquema del proyecto 41
2.2. Precio infraestructura IBM Cloud 43
2.3. Planificacion de tareas 45
2.4. Matriz de impacto de riesgos 48

3.1. Arquitectura de la solución propuesta 50
3.2. Modelo-Vista-Controlador aplicado 51
3.3. Arquitectura tecnológica del sistema 52
3.4. Flujograma esquemático global 53
3.5. Flujograma preparación entorno Cloud 54
3.6. Flujograma preparación entorno local 54
3.7. Flujograma detallado de ejecución estándar del sistema 55
3.8. Proceso entrenamiento del modelo 59
3.9. Patrón de detección facial 60
3.10. Codificación de una imagen 61
3.11. Imagen captura a través de la cámara web 64
3.12. Identificación facial detectada 66
3.13. Ejemplo fichero JSON 66
3.14. Flujo de proceso de Hyperledger Fabric 68
3.15. Instalación Hyperledger Fabric 71
3.16. Arquitectura Hyperledger 72
3.17. Generación de la red Hyperledger 79
3.18. Despliegue y lanzamiento de la red Hyperledger 80
3.19. Comprobación de la red en Hyperledger Composer Playground 80
3.20. Hyperledger Composer API-REST 81
3.21. Red consultada en Hyperledger Composer Playground 82
3.22. Escritura de la identificación en Hyperledger Fabric 84
3.23. Consulta del registro de identificación en Hyperledger Composer Playground 84

Índice de tablas

2.1. Presupuesto de personal 42
2.2. Presupuesto total 43
2.3. Niveles de ocurrencia 47
2.4. Niveles de impacto 47
2.5. Determinación del riesgo del proyecto 48

4.1. Caso de prueba 1 88
4.2. Caso de prueba 2 88
4.3. Caso de prueba 3 89
4.4. Caso de prueba 4 90
4.5. Caso de prueba 5 91
4.6. Caso de prueba 6 92
4.7. Caso de prueba 7 93
4.8. Caso de prueba 8 93
4.9. Caso de prueba 9 94
4.10. Caso de prueba 10 95
4.11. Caso de prueba 11 95
4.12. Caso de prueba 12 96
4.13. Caso de prueba 13 96

Glosario de términos

Machine Learning: disciplina científica del ámbito de la Inteligencia Artificial que crea sistemas de aprendizaje automático.

Deep Learning: técnica del Machine Learning, es un algoritmo automático estructurado que emula el aprendizaje humano con el fin de obtener ciertos conocimientos.

Dataset: Conjunto de datos de una tipología determinada.

Cloud: paradigma que permite ofrecer servicios de computación a través de una red, que normalmente es Internet.

Blockchain: tipo de tecnología de registro distribuido.

Ledger: término referido al libro de registros de una arquitectura de registro distribuido.

Trazabilidad: referida a la información, el término indica el registro e identificación de cualquier acción realizada sobre los datos.

Integridad: referida a la información, es la condición de mantener los datos sin alteraciones desde el momento de su almacenamiento.

Hyperledger: consorcio colaborativo de código abierto que tiene como objetivo la investigación sobre la tecnología de registro distribuido.

Hash: algoritmo para crear a partir de una entrada, una salida alfanumérica de longitud fija que representa un resumen de toda la información que se le ha proporcionado.

Convolución: función que, de manera línea y continua, transforma una señal de entrada en una nueva señal de salida.

Smart Contract: acuerdo entre dos o más partes dentro de una red Blockchain.

Chaincode: equivalente al término Smart Contract.

GitHub: sitio web que permite a los desarrolladores el almacenamiento y la administración de su código fuente.

Prueba de concepto: implementación, resumida o incompleta, de una idea de producto realizada con el propósito de verificar que el concepto es válido y puede ser industrializado.

GPU: unidad de procesamiento de gráficos. Se trata de un componente muy parecido a la CPU, solo que el tipo de procesamiento al que se dedica es procesamiento gráfico.

Support Vector Machine (SVM): algoritmo de aprendizaje automático supervisado que se puede emplear para clasificación binaria o regresión.

Principal Component Analysis (PCA): técnica de reducción de dimensionalidad que tiene como objetivo principal la extracción de toda la información de un conjunto de datos en pocas variables no correlacionadas entre sí.

Fisher Lineal Discriminant Analysis (FLDA): método de clasificación supervisado de variables cualitativas en el que dos o más grupos son conocidos a priori y nuevas observaciones se clasifican en uno de ellos en función de sus características.

Red Neuronal Recurrente (RNN): red neuronal que integra bucles de retroalimentación, permitiendo que la información persista durante algunas épocas de entrenamiento.

Deep Boltzman Machine (DBM): tipo de red neuronal recurrente, en la que todas las neuronas están conectadas entre sí y la información fluye en todas las direcciones..

Unidad lineal rectificada (Relu): función matemática que transforma los valores introducidos anulando los valores negativos y dejando los positivos tal y como entran.

Data Augmentation: técnica que permite ampliar un conjunto de datos a través de la introducción de perturbaciones o del uso de diferentes distribuciones de datos.

Distributed Ledger Technology (DLT): sistema en el que los registros o los datos son compartidos, accesibles y usados por todos los miembros de una red.

Árbol de Merkel: estructura de datos en árbol en el que cada nodo que no es una hoja está etiquetado con el hash de la concatenación de las etiquetas o valores de sus nodos hijo.

Bitcoin (BTC): se refiere a la criptodivisa del mismo nombre utilizada en Bitcoin.

Ether (ETH): se refiere a la criptodivisa de Ethereum.

Infrastructure as a Service (IaaS): método para ofrecer funcionalidades de computación, almacenamiento, redes y de otros tipos a través de Internet.

Internet of Things (IoT): agrupación e interconexión de dispositivos y objetos a través de una red dónde todos ellos podrían ser visibles e interaccionar.

Project Management Body of Knowledge (PMBOK): se refiere a la guía de los fundamentos para la dirección de proyectos del Project Management Institute.

JSON: formato de texto sencillo para el intercambio de datos.

Reglamento General de Protección de Datos (GDPR): marco legal en la Unión Europea que reemplaza a la Directiva de Protección de Datos. Busca proteger los datos personales y la forma en la que las organizaciones los procesan, almacenan y, finalmente, destruyen, cuando esos datos ya no son requeridos.

Self Sovereignty Identity (SSI): concepto de identidad soberana. Trata de asegurar que el usuario pueda: recuperar la propiedad de sus datos, que sirven como su identidad; así como controlar quién tiene acceso a cualquier información específica, qué parte de su información se comparte y cancelar el acceso cuando lo desee.

Introducción

El desarrollo de la inteligencia artificial (AI) y de las técnicas de aprendizaje automático (Machine Learning) está provocando un importante cambio en la sociedad que se ha visto reflejado, desde hace ya algunos años y cada vez con una mayor presencia, a nivel global. En la actualidad, la interacción de los seres humanos con la tecnología ha cambiado drásticamente y sigue en pleno proceso de cambio.

Tanto el propio desarrollo de la tecnología como los grandes avances técnicos están creciendo a pasos agigantados y están permitiendo obtener auténticos hitos dentro de este campo; como, por ejemplo, capacidades de cómputo o de almacenamiento impensables años atrás. Una evolución que influye de manera directa y clara en algunos ámbitos de reciente aparición como el Big Data, el Deep Learning, Blockchain o el Cloud Computing, entre otros.

La irrupción de estos nuevos campos de trabajo tiene como objetivo facilitar la vida diaria de las personas y aportarles, con la menor dificultad posible, nuevas funcionalidades que les permitan abordar, no sólo los problemas a los que se enfrenta la sociedad moderna en general, sino también los problemas más cotidianos, aquellos con los que tienen lidiar en su día a día.

El mundo empresarial, a su vez, también se ve impactado por el desarrollo de estas tecnologías, teniendo que adaptar su negocio y los procesos existentes a las demandas actuales del mercado.

Planteamiento del problema y motivación

Uno de los problemas que más controversia ha generado en la sociedad a lo largo del tiempo ha sido, sin duda, la identificación de las personas. En realidad, el estudio de la identidad se remonta a miles de años atrás y se ha sabido mantener a lo largo del tiempo hasta la actualidad. El ser humano siempre ha tratado de distinguir a los individuos que conforman la sociedad en la que viven, con el objetivo principal

de caracterizarlos y determinar los patrones que hacen que cada uno de ellos sean únicos.

Así, con el desarrollo tecnológico del que hablábamos en el punto anterior, han surgido nuevos sistemas de identificación que permiten realizar esta caracterización de personas de una manera cada vez mucho más exacta y simple. La aplicación y el uso de estos sistemas es de diversa índole y es, además, un proceso básico para cualquier acción que se realiza en la sociedad actual.

Una de las muchas aplicaciones de estos sistemas de identificación se encuentra en el ámbito empresarial. Algunos de ellos, como pueden ser las tarjetas identificativas (que ya llevan mucho tiempo entre nosotros), se utilizan para permitir o denegar el acceso a los empleados a determinados recursos restringidos o, por otro lado, para permitirles la entrada diaria a sus puestos de trabajo. Además, los sistemas de identificación también se utilizan diariamente en el ámbito personal: acceso con contraseñas a aplicativos web, identificación biométrica en los dispositivos móviles, etc.

Durante el año 2019, hemos visto cómo el Gobierno de España aprobaba una polémica ley -Real Decreto-Ley 8/2019-, por la que nace la obligación para toda empresa de registrar, de manera diaria y sin excepción, la jornada laboral de sus trabajadores. Tras su aprobación, la mayoría de las entidades han detectado una importante deficiencia en sus procesos de control e identificación, al no estar preparados para realizar este tipo de funciones. Así, la implantación de un sistema de identificación se convierte ahora en una tarea prioritaria y esencial para todas ellas.

En este documento se analizará el desarrollo de un sistema de identificación basado en técnicas de reconocimiento facial, que pueda ser utilizado como base para implementar ese obligatorio registro de la jornada laboral en las empresas. Un sistema que también podría ser de aplicación en otros ámbitos, como la gestión de acceso a un entorno controlado.

Objetivos del proyecto

El objetivo principal del proyecto es el **desarrollo de un sistema de identificación mediante técnicas de reconocimiento facial**. Para ello es necesario cumplir con los siguientes objetivos:

- Comprender cuáles son las técnicas más utilizadas de Machine Learning y todas sus tipologías.

- Conocer cuál es la diferencia entre las técnicas de Machine Learning y las propias de Deep Learning.
- Revisar el estado actual de los sistemas de identificación.
- Analizar los sistemas de identificación actuales basados en técnicas de reconocimiento facial e identificar sus ventajas e inconvenientes.
- Aprender cómo explotar un modelo de reconocimiento facial basado en Deep Learning.
- Comprender el concepto de Blockchain y todo su ecosistema.
- Estudiar el estado actual de la tecnología Blockchain y sus principales implementaciones.
- Saber qué es Hyperledger.
- Aprender a desplegar una red Blockchain privada y permisionada.
- Desarrollar un sistema global que incorpore el uso de ambas tecnologías, de tal manera que la tecnología Blockchain permita almacenar los registros de identificación de cada individuo, persistiendo en la cadena de bloques: un hash de la imagen, la etiqueta de la persona detectada y el timestamp del momento de la detección.

Metodología

Para el desarrollo de este proyecto, se ha seguido una metodología que parte de la definición del objetivo principal que se quiere conseguir y sigue los siguientes pasos:

1. Determinación del objetivo principal del proyecto.
2. Revisión del estado del arte.
3. Diseño de la solución a desarrollar.
4. Elección y explotación de un modelo de identificación basado en técnicas de reconocimiento facial.
5. Desarrollo e implementación de una Blockchain para almacenamiento de la información.
6. Consolidación de las dos partes principales del proyecto, que se han definido ya en los puntos anteriores.

7. Evaluación y pruebas del producto final.
8. Elaboración de conclusiones y consideraciones finales.

Estructura del trabajo

Siguiendo la metodología del punto anterior, el presente documento se divide en cinco capítulos, para conseguir un acercamiento más sencillo al proyecto:

- **Introducción**: se contextualiza el proyecto dentro de la sociedad actual, explicando el planteamiento del problema y su motivación, además del objetivo principal y la metodología y herramientas empleadas para su implementación.
- **Capítulo 1 - Revisión del estado del arte**: contextualiza, revisa y expone un análisis con posibles planteamientos alternativos a problemas similares.
- **Capítulo 2 - Desarrollo de la solución**: proporciona el diseño realizado para abordar el problema descrito y planteado como objetivo principal del proyecto, junto a la arquitectura que debe poseer el sistema para llegar a los resultados deseados. Adicionalmente, este capítulo detalla el proceso de la implementación del proyecto.
- **Capítulo 3 - Evaluación y pruebas de la solución**: describe la manera de evaluar el proyecto y los casos de prueba ejecutados para garantizar su máxima calidad.
- **Capítulo 4 - Líneas futura**s: analiza las posibles líneas futuras de trabajo que se pueden abordar en relación al proyecto.
- **Capítulo 5 - Conclusiones**: presenta un resumen final del trabajo realizado con los resultados principales más determinantes.

Capítulo 1

Revisión del estado del arte

Conocer los diferentes sistemas de identificación, junto a las principales técnicas de reconocimiento facial, es esencial para comprender el objetivo principal de este proyecto y el contexto que lo engloba. Además, no deja de tener menor importancia la comprensión de los fundamentos básicos de Blockchain, tecnología sobre la cual se basa la trazabilidad e integridad de la información almacenada de la solución presentada en este proyecto. Por ello, en este capítulo se realiza un breve estudio sobre ambas tecnologías, con el fin de que el lector pueda adquirir un conocimiento general de las mismas que le permita comprender el proyecto en su totalidad.

1.1. Sistemas de identificación y técnicas de reconocimiento

La capacidad de probar que una persona es quien dice ser es un componente fundamental en el desarrollo social, dado que es, entre otras cosas, una pieza clave para poder acceder a cualquier tipo de servicio. Así, la identidad permite acceder a servicios públicos o privados, como la sanidad, educación, servicios financieros, etc.

Si bien el hecho de poder identificar a una persona parece algo sencillo y trivial, en realidad, es un problema reconocido a nivel mundial. En un estudio realizado en el año 2018, el Banco Mundial estimó que existían alrededor de 1.8 billones de personas que no son capaces de demostrar su identidad [1]. La mayoría de estas personas se corresponde con personas residentes en Asia o África. Aprovechando la transformación digital y la implantación de soluciones de identificación más modernas, el Banco Mundial ha visto una potencial oportunidad en la resolución de esta

problemática, sustituyendo los métodos de identificación tradicionales por sistemas de identificación más robustos y eficientes que, hasta ahora, no habían sido posibles.

Por otro lado, existe un objetivo global incluido en el documento de Naciones Unidas «Sustainable Development Goal (SDG)» en el que se indica que en el año 2030 se debe haber conseguido que todas las personas del mundo dispongan de identificación legal [2].

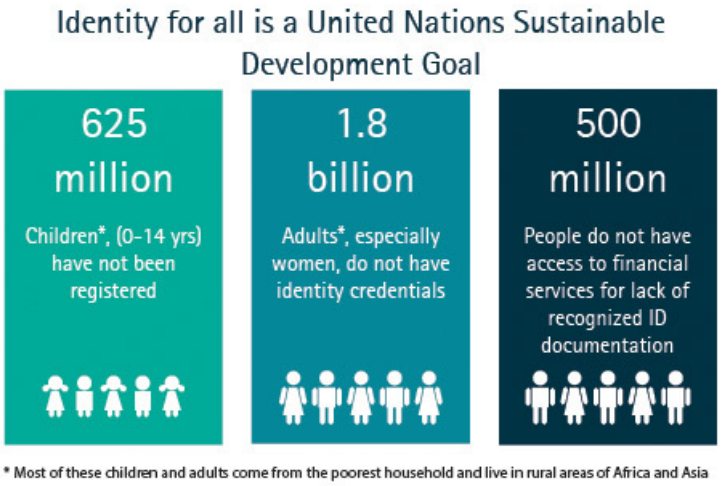


Figura 1.1: Datos de Naciones Unidas sobre la identidad global

Fuente: Naciones Unidas

Tradicionalmente, los sistemas de identificación han estado basados en interacciones o documentos físicos. Las técnicas que se han venido utilizando se han desarrollado a través de dos enfoques diferentes: identificación basada en conocimiento e identificación basada en posesión. Sin embargo, con el desarrollo de la tecnología han surgido los sistemas de identificación biométricos, que suponen una evolución clara sobre los enfoques tradicionales [3].

La siguiente imagen muestra un esquema de los diferentes sistemas de identificación que serán tratados en este capítulo:

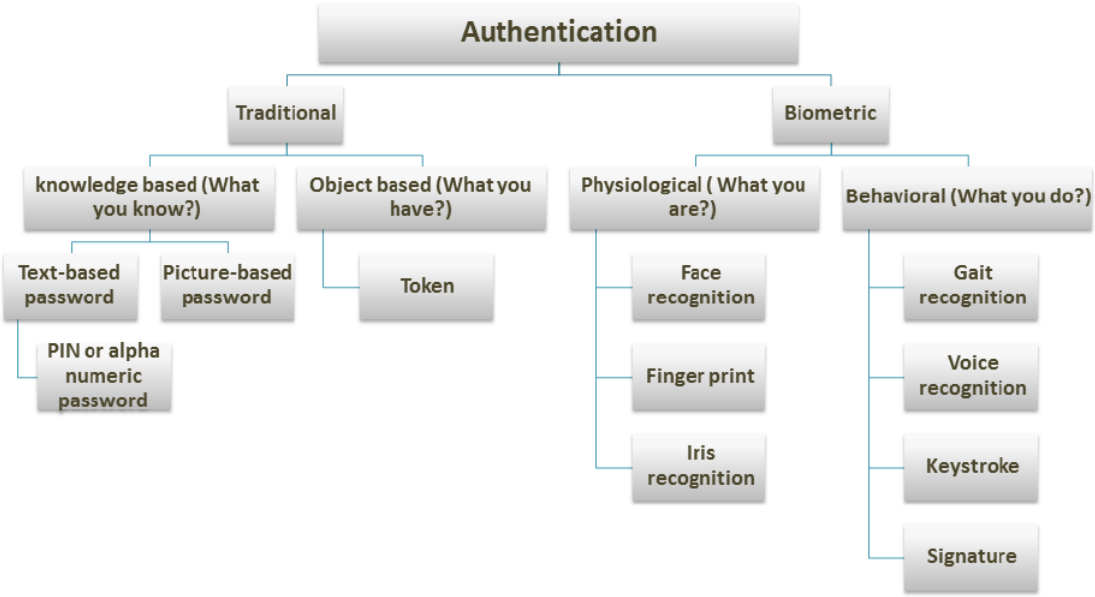


Figura 1.2: Topología de los sistemas de identificación

Fuente: Biometric and Traditional Mobile Authentication Techniques [3]

Identificación basada en conocimiento

Las técnicas de identificación basadas en conocimiento son las más utilizadas. Parten de la premisa de "qué conoce el usuario" que le identifique, e incluyen dos clases:

- **Contraseñas basadas en texto**: incluye números de identificación personal (PIN) y contraseñas alfanuméricas. Ambas presentan problemas de seguridad ya que los usuarios utilizan, normalmente, contraseñas débiles y no las cambian, por lo general, con la frecuencia deseada. Además, las contraseñas no representan a un usuario, por lo que un robo de contraseña provoca un robo de identidad [4].
- **Contraseñas basadas en imágenes**: este método se basa en memorizar imágenes en lugar de textos, una técnica que se hizo muy popular debido a su uso en los dispositivos móviles Android, en los que el usuario debe introducir un patrón para acceder a su dispositivo. Pueden incrementar en parte la seguridad del método anterior, pero la contraseña sigue sin representar al individuo.

Identificación basada en posesión

Dado que, como se ha visto, los sistemas de identificación basados en conocimiento presentan vulnerabilidades, se desarrollaron las técnicas de identificación basadas en posesión u objetos. Éstas se utilizan normalmente como un segundo factor de autenticación e incluyen la identificación basada en tokens o claves One-Time Password (OTP) [5]. La principal desventaja de estos sistemas es que, normalmente, requieren de un segundo dispositivo donde recibir el token o la contraseña por lo que, en caso de no disponer de dicho dispositivo no se podría utilizar esta técnica.

Identificación biométrica

Sin embargo, los sistemas basados en los enfoques definidos anteriormente - conocimiento y posesión - no satisfacen los requerimientos de seguridad deseados. En ambos, es imposible diferenciar un individuo genuino de un impostor que de manera fraudulenta adquiera los mismos privilegios que la persona real. Es ante esta falta de representación única por lo que surge la identificación biométrica.

La palabra biometría procede de dos palabras griegas: bios (vida) y metron (medida). El origen de la palabra define con exactitud el término, el cual se refiere a la identificación de un individuo basándose en sus características fisiológicas o comportamentales. Siguiendo con la analogía de los apartados anteriores, la biometría se clasifica como un proceso de identificación basado en características personales. Así, la identificación biométrica es un proceso en el cual un usuario es reconocido automáticamente a partir de un vector de características extraído de su comportamiento o fisiología.

Lin Hong y Anil Jain hacen un análisis sobre los requerimientos esenciales de un sistema de identificación biométrico en su artículo «Integrating Faces and Fingerprints for Personal Identification» [6], en el que afirman que cualquier característica humana puede ser utilizada en los procesos biométricos para realizar una identificación personal, siempre y cuando satisfaga las siguientes condiciones:

- **Universalidad**: todas las personas deben tener dicha característica.
- **Unicidad**: no pueden existir dos personas iguales basándose en dicha característica.
- **Inmutabilidad**: la característica no puede ser modificada.
- **Recopilación**: la característica debe poder ser extraída de las personas y medida de manera exacta.

Sin embargo, no todas las características biométricas que satisfacen las condiciones anteriores son factibles en un sistema biométrico práctico, dado que hay otros elementos a tener en cuenta:

- **Rendimiento**: el sistema debe ser preciso, rápido y robusto.
- **Aceptabilidad**: la sociedad debe aceptar compartir una característica biométrica única.
- **Suplantación**: refleja que el sistema no debe ser fácilmente engañado mediante métodos fraudulentos.

Por lo tanto, un sistema biométrico práctico debe ser capaz de:

1. Obtener un rendimiento adecuado en la identificación de un individuo y una velocidad razonable.
2. No ser dañino para los usuarios y debe ser aceptado por la sociedad.
3. Ser lo suficientemente robusto ante métodos fraudulentos.

Volviendo a las características biométricas, es importante destacar que, a raíz del comportamiento de un individuo, existen dos tipologías diferenciadas [3]:

- **Fisiológico**. Depende de aquellas características que un individuo realmente posee. A continuación, se detallan las técnicas más utilizadas en la actualidad [7]:
    - **Reconocimiento facial**: método por el cual una cara humana es capturada para, posteriormente, ser utilizada con el fin de identificar a un individuo. Estos métodos miden y analizan la estructura y la forma de la cara, sus proporciones (distancia entre los ojos, nariz y boca; el contorno de las cuencas de los ojos; los lados de la boca; la localización de la nariz y la boca; etc.). En la verificación de identidad, el usuario debe situarse frente a una cámara. Para prevenir un posible ataque mediante el uso de una imagen en lugar del individuo real, estos sistemas, en ocasiones, esperan el parpadeo de los ojos, solicitan que el usuario sonría, haga algún gesto o utilizan otra técnica con la que se comprueba la temperatura de la cara: la termografía facial.

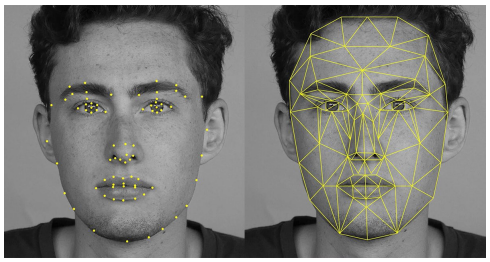


Figura 1.3: Reconocimiento facial

Fuente: `http://www.liberaldictionary.com/face-recognition/top-10-facial-recognition-apis-updated-for-2018-rapidapi/`

- **Huella**: es el método de identificación más antiguo. Se captura la huella de un usuario y se contrasta contra una huella almacenada previamente. Las huellas tienen pequeños surcos en la piel en forma de líneas que forman patrones únicos. Estos patrones pueden ser continuos o discontinuos y son la base del reconocimiento dactilar.

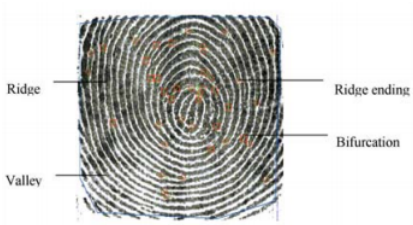


Figura 1.4: Reconocimiento huella

Fuente: An Efficient Approach for Robust Fingerprint Recognition System [8]

- **Reconocimiento de iris**: el iris posee un patrón muy complejo que puede contener muchas características únicas como pueden ser surcos, anillos, coronas, etc. El escaneo de iris es menos intrusivo que el de la retina, ya que éste es visible desde varios metros de distancia. Adicionalmente, los cambios en el iris debidos a la luz pueden proporcionar una verificación secundaria que garantiza que el iris es de una persona que se encuentra viva.

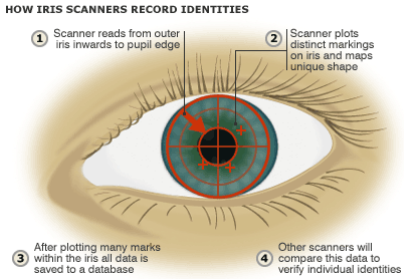


Figura 1.5: Reconocimiento iris

Fuente: `https://medium.com/mawi-band/your-heart-is-a-new-biometrics-eda2c168f7b`

- **Escáner de retina**: el reconocimiento de retina crea una firma con el ojo a partir de las características vasculares de ésta, que son únicas en cada individuo y en cada ojo. Es uno de los métodos biométricos más seguros, pero es un método intrusivo y puede revelar determinadas condiciones médicas que hagan que la sociedad se cuestione su uso.

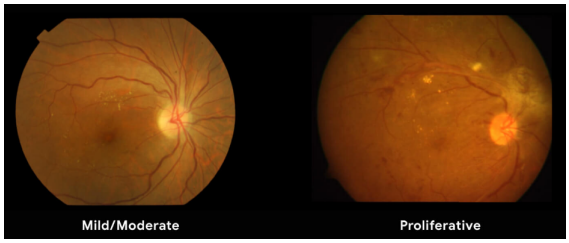


Figura 1.6: Reconocimiento de retina

Fuente: `http://www.m2sys.com/blog/biometric-hardware/`

`iris-recognition-vs-retina-scanning-what-are-the-differences/`

- **Geometría de la mano**: utiliza la forma geométrica de la mano para identificar a un usuario. Es un método cuestionable, ya que estas características no describen de manera suficientemente unívoca a un individuo.

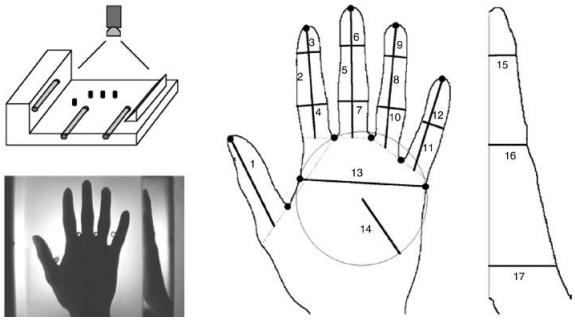


Figura 1.7: Reconocimiento basado en geometría de mano

Fuente: `https://link.springer.com/content/pdf/10.1007/978-3-642-27733-7_251-2.pdf`

- **Comportamental**. Depende de las características de comportamiento de un individuo. A continuación, se detallan las técnicas más utilizadas [7]:

  - **Reconocimiento del paso**: es una técnica de reconocimiento biométrico que utiliza una secuencia de imágenes a partir de un vídeo de una persona caminando para identificar sus movimientos. En principio, no es un método que permita realizar demasiada distinción entre individuos y, además, el paso es un comportamiento que puede verse modificado en el tiempo debido a los cambios físicos que se puede producir en el cuerpo, por ejemplo. Pese a no ser una técnica intrusiva, su uso no está demasiado extendido debido a su eficacia.

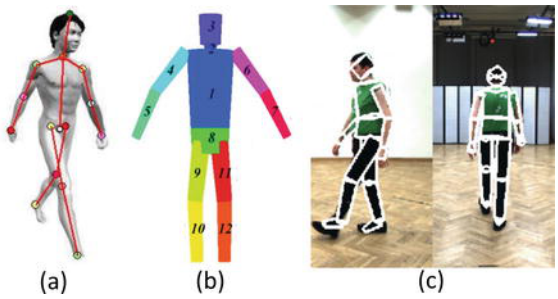


Figura 1.8: Reconocimiento del paso

Fuente: `https://www.intechopen.com/books/motion-tracking-and-gesture-recognition/gait-recognition`

  - **Reconocimiento de voz**: los sistemas de reconocimiento de voz utilizan, en la mayoría de los casos, un micrófono para almacenar la voz enunciando un texto conocido. Esta grabación se utiliza para extraer patrones únicos y formar una plantilla de voz, que puede ser de dos tipos: estocástica (requieren coincidencia probabilística) y plantilla de modelo (utilizan patrones deterministas). Estos sistemas tienen un grado de aceptación alto dado que no son intrusivos y su precisión es bastante elevada.

Figura 1.9: Reconocimiento de voz

Fuente:

`https://searchcustomerexperience.techtarget.com/definition/voice-recognition-speaker-recognition`

- **Pulsación de teclas**: es una técnica desarrollada para mejorar la identificación mediante texto y está basada en la extracción de características durante la pulsación de teclas (por ejemplo: tiempo de pulsación de tecla o intervalo de tiempo entre dos pulsaciones).

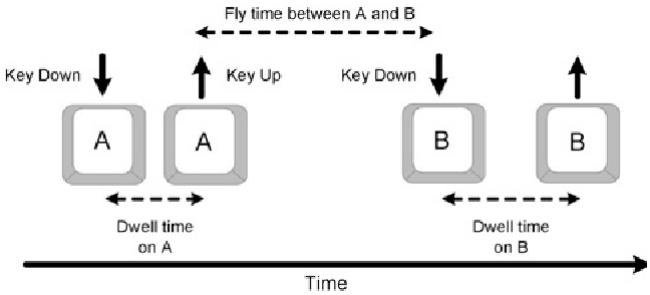


Figura 1.10: Pulsación de teclas

Fuente: `https://blog.daftcode.pl/building-supervised-models-for-user-verification-part-1-of-the-tutorial-7496d5d394b9`

- **Reconocimiento de firma**: La firma manuscrita es un rasgo biométrico que cuenta con una alta aceptación social por parte de los usuarios, habiendo sido utilizada durante siglos como método de autenticación de documentos legales y de transacciones y contratos civiles. Existe gran dificultad para modelar la firma, por lo que se necesita un gran volumen de datos y existe gran escasez de los mismos (debido a la alta variabilidad intra-usuario). Además, es difícil detectar firmas falsas, dado que el grado de bondad de un impostor no se puede predecir (baja variabilidad inter-usuario). El reconocimiento de firma se puede realizar de manera estática (cuando la firma ya se ha realizado) o de manera dinámica (en el momento de realización de la firma).

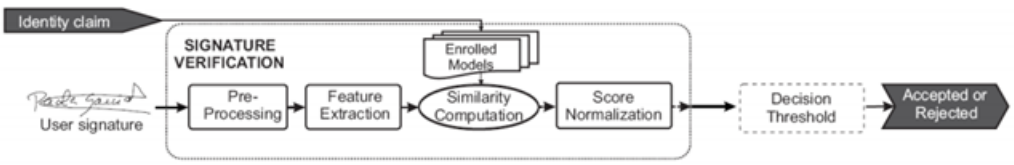


Figura 1.11: Reconocimiento de firma

Fuente: Curso Universidad Autónoma de Madrid

1.2. Técnicas tradicionales de Machine Learning

El Machine Learning (ML), o aprendizaje automático, nació en la década de los años 60 como una idea de la inteligencia artificial. El origen del Machine Learning pretendía analizar el reconocimiento de patrones en los procesos de ingeniería y matemáticas y el aprendizaje de los mismos por parte de los ordenadores. El objetivo fundamental era conseguir que los ordenadores aprendieran únicamente basándose en los datos.

Un proceso de Machine Learning trata de encontrar una transformación algorítmica parametrizada mediante una entrada y una salida que minimice una medida de error, ajustando de manera iterativa los parámetros de dicha transformación. Este proceso se realiza con el objetivo de descubrir algoritmos que consigan obtener conclusiones derivadas de un conjunto de datos sin que el ser humano tenga que desarrollar un programa para ello [9].

A la hora de analizar un proceso de ML existen una serie de métricas que permiten evaluar su eficacia [10]:

- **Matriz de confusión**: tabla que compara los resultados predichos de un modelo de ML contra los datos reales. Las filas representan los valores reales y las columnas los predichos. Se pueden obtener falsos negativos/positivos (FN o FP) y verdaderos negativos/positivos (TP o TN).

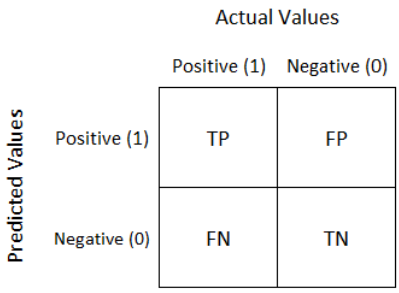


Figura 1.12: Matriz de confusión

Fuente: https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62

- **Precisión**: representa el porcentaje total de elementos predichos correctamente. Se calcula como (TP + TN) / (TP + FP + TN + FN).
- **Sensibilidad**: hace referencia al número total de elementos identificados correctamente como positivos sobre el total de positivos reales. Se calcula como TP / (TP + FN).

- **Área bajo la curva de funcionamiento del receptor (ROC)**: gráfico que muestra el rendimiento de un modelo de clasificación. Representa la tasa de verdaderos positivos en el eje Y y la tasa de falsos positivos en el eje X.

Existen gran cantidad de técnicas de ML que han sido desarrolladas en los últimos años, focalizando su principal diferencia en los algoritmos utilizados por cada una y en la representación del flujo de datos sobre el modelo.

El estudio de los algoritmos de Machine Learning implica el conocimiento de ciertas características acerca del problema objeto de estudio, que deben ser analizadas antes de tratar de resolverlo. En primer lugar, es importante conocer el problema que se está tratando. Los problemas que se pueden resolver mediante técnicas de ML se pueden clasificar en:

- **Clasificación**: realiza una categorización en el conjunto de datos. La variable de salida toma valores basados en etiquetas de clases.
- **Regresión**: predice un valor sobre el conjunto de datos. La variable de salida toma valores continuos.

En segundo lugar, hay que tener en cuenta la elección de los datos que alimentan el algoritmo de ML. Este proceso es muy importante dado que el éxito de solventar el problema está muy relacionado con la elección del algoritmo. Dicha elección depende, sobre todo, de si el conjunto de datos contiene información sobre la variable de salida o no.

Los problemas de Machine Learning pueden ser clasificados en: aprendizaje supervisado o no supervisado:

- **Aprendizaje supervisado**. Los algoritmos están basados en el entrenamiento de los datos desde un conjunto de datos inicial que incluye una correcta clasificación o valor para cada caso. Este entrenamiento permite que este tipo de modelos puedan categorizar nuevos datos sin conocer su clasificación correcta. Para este tipo de enfoque, hay que asumir la existencia de un supervisor que asigne el valor correcto para cada uno de los ejemplos del conjunto de datos inicial.
- **Aprendizaje no supervisado**. Los algoritmos están basados en identificar patrones ocultos en datos de entrada no etiquetados.

La resolución del problema para la detección facial se enmarca dentro de la categoría de clasificación mediante aprendizaje supervisado. Los algoritmos más utilizados

de Machine Learning para afrontar el problema de la detección facial, tradicionalmente, han sido las máquinas de soporte vectorial (SVM'S) y el análisis de componentes principales (PCA). Por tanto, a continuación se realiza una definición de ambos.

Máquinas de soporte vectorial (SVM's)

Las máquinas de soporte vectorial o Support Vector Machines (SVM) son clasificadores derivados de la teoría de aprendizaje estadístico publicada por Vapnik y Chervonenkis [11]. Se trata de un conjunto de algoritmos de aprendizaje supervisado y se pueden utilizar tanto para problemas de clasificación como de regresión.

De manera resumida, una SVM es un modelo de aprendizaje supervisado que representa los puntos de muestra en el espacio, separando las clases en subespacios lo más grandes posibles mediante un hiperplano de separación denominado vector de soporte.

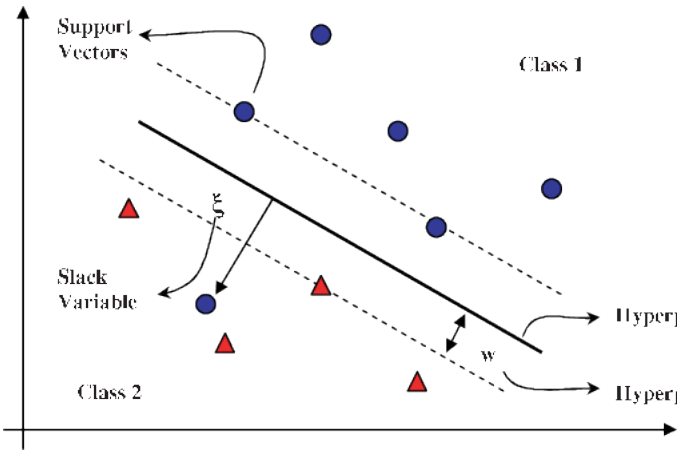


Figura 1.13: SVM

Fuente: `https://www.researchgate.net/figure/`

`The-soft-margin-SVM-classifier-with-slack-variables-x-and-support-vectors-shown_fig2_220606206`

SVM se ha convertido en un método popular porque tiene gran capacidad para la resolución de problemas de clasificación no lineales con muy buenos resultados.

Para el problema concreto de la detección facial, se aplican las SVM en dos fases: de manera inicial se plantea el problema para detectar si en una imagen existe una cara o no; posteriormente, se extiende el resultado para tratar un problema multiclase, como se verá más adelante.

Análisis de componentes principales (PCA)

El análisis de componentes principales o PCA es un método estadístico simple que se utiliza para reducir la dimensionalidad a la vez que se minimiza el error cuadrático medio de reconstrucción. Es una técnica utilizada para describir un conjunto de datos en nuevas variables (componentes) no correlacionadas entre sí [12].

El algoritmo de PCA busca una proyección de las variables, según la cual los datos queden representados de la manera más óptima en términos de mínimos cuadrados. Esto hace que un conjunto de observaciones de variables correlacionadas entre sí, se convierta en un nuevo conjunto de valores sin correlación lineal, y es lo que se denomina componentes principales.

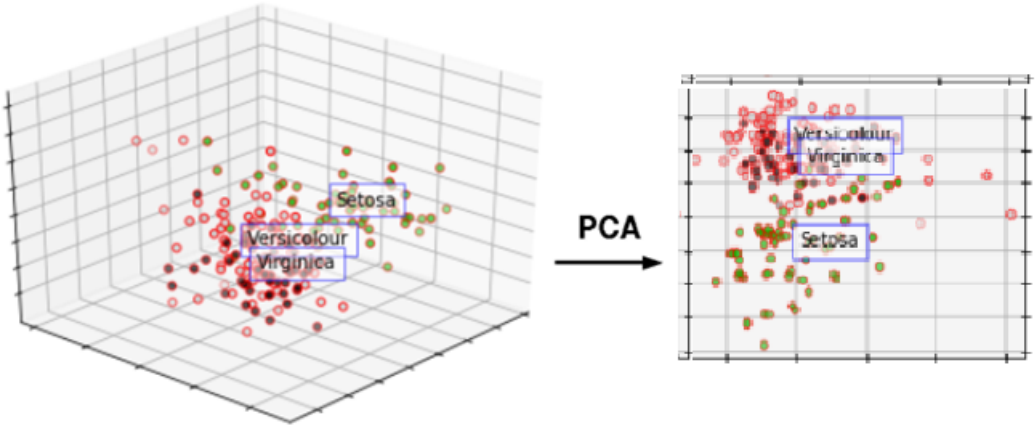


Figura 1.14: PCA

Fuente: `https://www.deeplearningitalia.com/metodos-para-la-reduccion-de-la-dimensionalidad-manifold-based-el-caso-isomap/`

Aplicación sobre la detección facial

Existen diversos estudios en los que se demuestra y se comprueba la efectividad de los algoritmos anteriores para resolver la problemática de la detección facial [13], [14], [15] .

El objetivo principal del reconocimiento facial es distinguir la señal de entrada (en forma de imagen) de un número variado de clases (personas). La señal de entrada suele tener ruido debido a diferencias en la luz, en la postura, etc. Por otro lado, cada cara tiene diferentes características pero, por lo general, se puede detectar un patrón similar en ellas (por ejemplo, observando la nariz, los ojos, la boca y la distancia entre dichos elementos). El reconocimiento facial basado en estas características es conocido como Eigenface [13]. Los patrones pueden ser extraídos matemáticamente

utilizando PCA. El principio básico de estos métodos es obtener la información única de la cara, codificarla y compararla con los resultados decodificados previos.

Poon, Amin y Yan [14], analizan el rendimiento de este tipo de algoritmos. En su estudio concluyen que cuanto mayor es el número de características extraídas de una imagen, mayor es el porcentaje de acierto, lo cual no es muy sorprendente. No obstante, este incremento tiende a saturarse a partir de cierto número de características. De esta manera, consiguen obtener una precisión de alrededor del 90 % con un conjunto de datos de en torno a 400 imágenes extrayendo siete características de las imágenes.

Por otro lado, en situaciones en las que las imágenes tienen grandes variaciones en iluminación y en expresiones, resulta muy útil el método Fisherface [13]. Es una mejora del método Eigenface utilizando el análisis lineal discriminante de Fisher (FLDA). Este enfoque trata de encontrar la dirección de la proyección en la que la pertenencia de la imagen a determinadas clases presenta una separación máxima.

Por último, las SVM's tratan de encontrar un hiperplano que maximice el margen entre dos clases. Sani, Ishak y Samad [15] demuestran que utilizando dicho algoritmo como un clasificador multiclase se puede afrontar el problema. Dado que las SVM's están pensadas inicialmente para clasificar únicamente dos clases, es necesario realizar ciertas modificaciones para poder construir un clasificador multiclase. Existen dos técnicas principales: combinar múltiples clasificadores binarios o utilizar todos los datos en el algoritmo de optimización. Bajo este método se obtiene una precisión de un 74 % con 15 muestras analizadas.

Setiowati, Zulfanahri, Franita y Ardiyanto [13] presentan otra comparativa de precisión entre los métodos que se acaban de analizar, utilizando la base de datos de YALE [16] como entrada:

Método	**Accuracy**
Eigenface	77,9 %
Fisherface	85,2 %
SVM	90,6 %

En él se puede observar cómo las técnicas tradicionales de Machine Learning para la detección facial ofrecen ratios de precisión mejorables. Además, un problema de este tipo requiere de un porcentaje de acierto de, prácticamente, el 100 % dadas las implicaciones que puede tener identificar de manera errónea a un individuo.

1.3. Deep Learning

El Deep Learning (DL) es una técnica de Machine Learning que permite que los ordenadores resuelvan problemas naturales para las personas [17]. Es una técnica de aprendizaje mediante ejemplos. El DL es capaz de obtener un gran rendimiento con unas tasas de acierto que, en ocasiones, superan la precisión humana.

Las primeras teorías sobre el DL surgieron en la década de los años 80 [18]. Sin embargo, debido a las limitaciones tecnológicas existentes en esa época, las teorías no pudieron ser llevadas a la práctica. El avance tecnológico desarrollado en los últimos años está permitiendo que las aplicaciones de DL proliferen y sean una de las primeras opciones del mercado [19], [20]. Su auge se debe en gran parte a dos motivos:

- **Aumento de la capacidad de almacenamiento e irrupción de infraestructuras Big Data**. El DL requiere gran cantidad de datos.
- **Incremento de la capacidad de cómputo**. El DL requiere una potencia de cálculo muy elevada. La mayoría de las aplicaciones requieren de GPU's de alto rendimiento, que ofrecen una arquitectura paralela muy eficiente para los problemas de DL. Además, la irrupción de tecnologías cloud hace que la inversión en hardware no sea significativa y que se puedan hacer pruebas inmediatas en estos entornos.

La mayoría de los algoritmos de aprendizaje de DL utilizan arquitecturas de redes neuronales, que se introducen a continuación.

Redes neuronales

Una red neuronal es un paradigma de aprendizaje y procesamiento automático inspirado en el funcionamiento del sistema nervioso de los animales. Se trata de sistemas de interconexión de neuronas que colaboran entre sí para producir un estímulo de salida [21].

También se puede definir como un conjunto de neuronas interconectadas con organización jerárquica donde se procesa información a través de los estados dinámicos de las neuronas como respuesta a entradas externas.

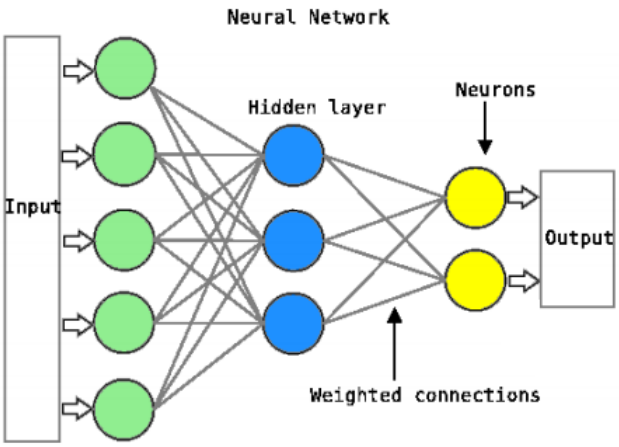


Figura 1.15: Red neuronal

Fuente:

https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networks-d7834f67a4f6

Las principales ventajas de las redes neuronales son:

- Poseen **aprendizaje adaptativo**: generan su distribución de pesos mediante el aprendizaje.
- Su arquitectura es **auto-organizativa**.
- Presentan **tolerancia a fallos**: reconocen patrones con ruido y las redes pueden seguir funcionando aunque se destruya parte de la misma.
- **Redundancia** de los datos: la información se encuentra distribuida en las conexiones neuronales.

Dos de los algoritmos más utilizados son: el perceptrón simple y el perceptrón multicapa (MLP). El perceptrón simple únicamente sirve para tratar problemas de clasificación mediante aprendizaje supervisado, mientras que el perceptrón multicapa trata problemas de clasificación y de regresión mediante aprendizaje supervisado.

Arquitectura de DL

El uso de arquitecturas de redes neuronales en DL hace que, en muchas ocasiones, sus modelos se denominen redes neuronales profundas. El DL también puede ser definido como el uso de redes neuronales profundas que están conectadas entre sí y que, mediante la aplicación de algoritmos de cálculo, obtienen una salida. Las capas conectadas directamente utilizan los resultados de la capa anterior como entrada para producir la siguiente salida:

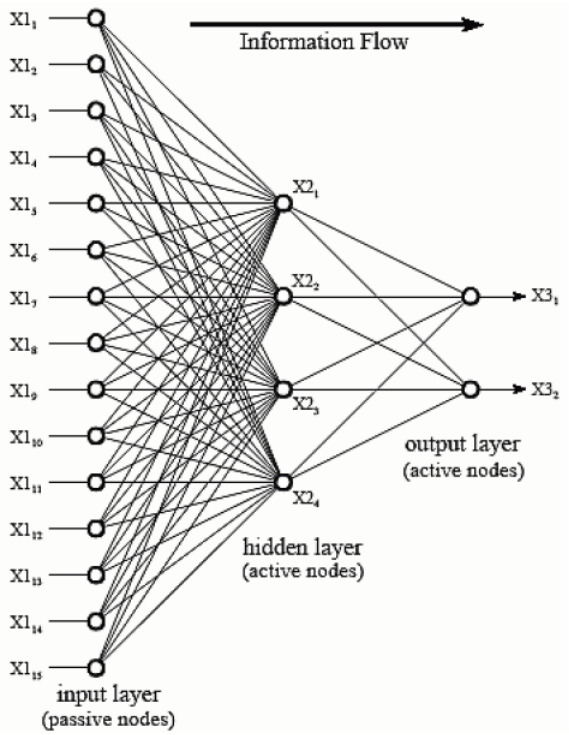


Figura 1.16: Estructura Deep Learning

Fuente: Neural Networks – State of Art [22]

El término profundo hace referencia a las capas ocultas de la red neuronal. La diferencia entre las redes neuronales tradicionales y las redes neuronales profundas radica en que, en las primeras, únicamente se tienen dos o tres capas ocultas, mientras que, en el segundo caso, se pueden tener hasta 150 capas.

Una de la ventaja de los modelos de DL es que se entrenan a partir del uso extenso de conjuntos de datos. Las arquitecturas de redes neuronales profundas aprenden directamente de los datos y no necesitan una extracción manual de características [23].

Li Deng presenta una clasificación de los tres tipos principales de arquitecturas de DL [24]:

1. **Arquitecturas profundas generativas**: destinadas a caracterizar las propiedades de correlación más elevadas de los datos observados para analizar los patrones y caracterizar las distribuciones estadísticas conjuntas de estos datos y sus clases asociadas. Ejemplos de estas arquitecturas son:

    - **Autoencoder**. Los autoencoders o autocodificadores son redes neuronales profundas cuyo objetivo principal es aprender a generar en la salida exactamente la misma información que recibe en la entrada. Esto se rea-

liza, en primer lugar, comprimiendo la entrada en un espacio de variables latentes y, posteriormente, reconstruyendo la salida en base a la información adquirida (las capas de entrada y de salida deben tener el mismo número de neuronas). La clave está en las capas ocultas, donde la red almacena la representación abstracta de los datos de entrenamiento.

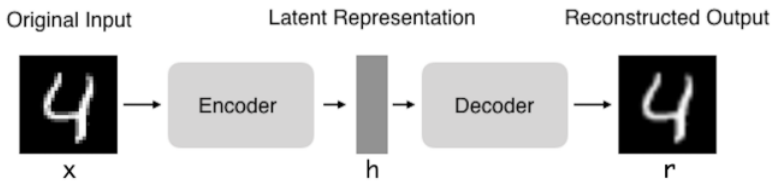


Figura 1.17: Autoencoder

Fuente:

`https://towardsdatascience.com/auto-encoder-what-is-it-and-what-is-it-used-for-part-1-3e5c6f017726`

- **Deep Boltzmann Machine (DBM)**. Una máquina de Boltzmann contiene numerosas capas con multitud de variables ocultas. La característica principal es que los nodos ocultos no se pueden conectar a otros nodos, y los nodos visibles tampoco.

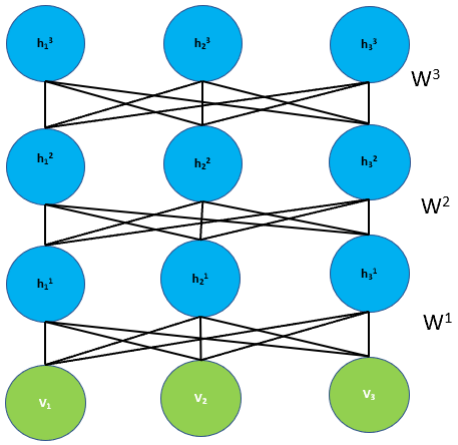


Figura 1.18: Deep Boltzmann Machine

Fuente: `https://medium.com/datadriveninvestor/deep-learning-deep-boltzmann-machine-dbm-e3253bb95d0f`

- **Redes neuronales recurrentes (RNN)**. Las RNN no tienen una estructura de capas definidas, sino que permiten conexiones entre neuronas. Este tipo de arquitectura puede llegar a crear ciclos en la red, lo que permite que esta red tenga memoria.

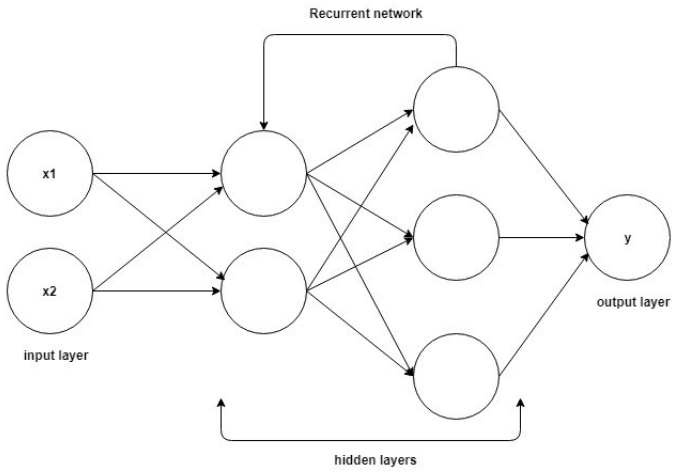


Figura 1.19: Red neuronal recurrente

Fuente: `https://hackernoon.com/rnn-or-recurrent-neural-network-for-noobs-a9afbb00e860`

2. **Arquitecturas profundas discriminativas**: destinadas a proporcionar directamente poder discriminativo para la clasificación de patrones, normalmente caracterizando las distribuciones posteriores de las clases condicionadas en los datos visibles. Ejemplos de estas arquitecturas son:

   - **Conditional Random Fields (CRF)**. Son modelos discriminativos utilizados para predecir secuencias. Utilizan información contextual de capas anteriores, incrementando la cantidad de información que posee el modelo para realizar la predicción.

   - **Redes neuronales convolucionales (CNN)**. Una CNN transforma las características aprendidas con los datos de entrada y emplea capas convolucionales 2D, lo cual hace que esta arquitectura resulte adecuada para procesar datos 2D como, por ejemplo, imágenes [23]. Este tipo de redes no necesitan una extracción manual de características, ya que realizan dicha extracción de manera automática a partir de las imágenes. Uno de los aspectos más determinantes es que las características más relevantes no se entrenan previamente, sino que se aprenden mientras la red se entrena con un conjunto de imágenes. Este procesamiento hace que los modelos de DL basados en esta arquitectura sean muy precisos para la detección facial.

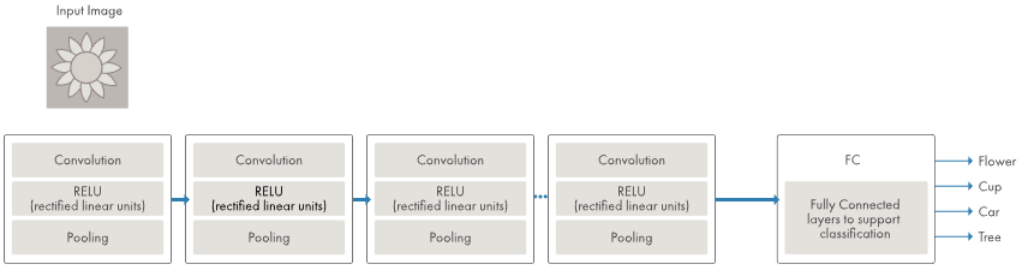


Figura 1.20: Estructura red neuronal convolucional

Fuente: MathWorks[23]

Para realizar la extracción automática de características, las CNN aprenden a diferenciarlas a través de cientos de capas ocultas. Cada capa aumenta la complejidad de las características aprendidas. Por ejemplo, la primera capa oculta puede aprender a detectar los bordes de una imagen, la segunda a detectar la forma de una cara y, así, sucesivamente.

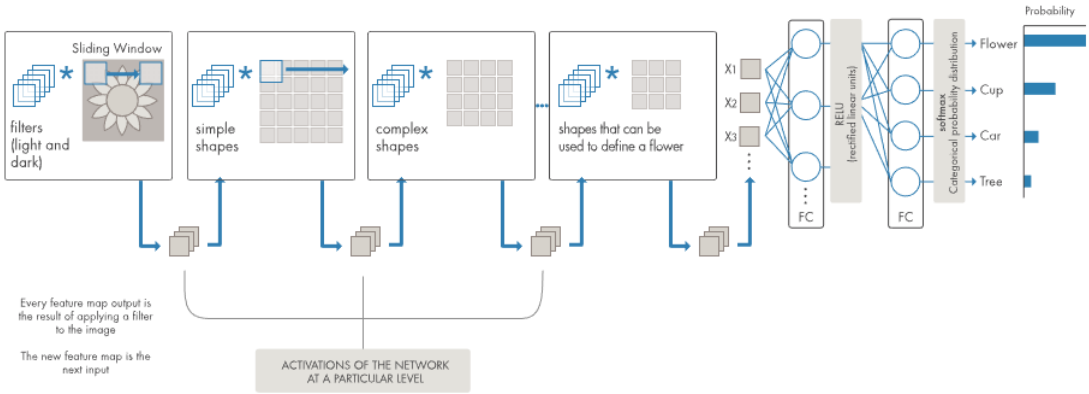


Figura 1.21: Ejemplo capas red neuronal convolucional

Fuente: MathWorks[23]

3. **Arquitecturas profundas híbridas**: su meta es la discriminación pero ésta es asistida con los resultados de arquitecturas generativas para una mejor optimización o regularización. Un ejemplo de estas arquitecturas sería cualquier tipo de combinación posible entre las dos anteriores.

1.4. Técnicas de reconocimiento facial basadas en Deep Learning

Tras haber realizado una revisión sobre qué es el Deep Learning y sus principales algoritmos, en esta sección se abordan las técnicas más utilizadas para afrontar el problema de la detección facial.

La mayoría de las técnicas utilizadas actualmente están basadas en redes neuronales convolucionales (CNN) o presentan arquitecturas híbridas. M. Wang y W. Deng presentan una revisión (ordenada cronológicamente) de las arquitecturas más utilizadas en el tratamiento del reconocimiento facial [25]:

1. **AlexNet** [26]. Surgida en la competición ILSVRC de 2012, consiguió los mejores resultados con un amplio margen. Esta técnica consiste en cinco capas convolucionales y tres capas totalmente conectadas, integrando, a su vez, varias técnicas como funciones ReLU y data augmentation.

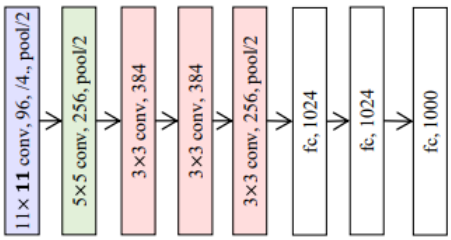


Figura 1.22: AlexNet

Fuente: Deep Face Recognition: A Survey [25]

2. **VGGNet** [27]. En 2014 VGGNet propone una red estándar que utiliza filtros convolucionales muy pequeños, pero que aumenta la profundidad de la red a 16-19 capas.

Figura 1.23: VGGNet

Fuente: Deep Face Recognition: A Survey [25]

3. **DeepFace** [28]. Fue la primera en utilizar una red convolucional de nueve capas con multitud de capas conectadas localmente.

4. **GoogleNet** [29]. En 2015, Google propuso una red de 22 capas, introduciendo un módulo de inicio con mapas de características híbridas.

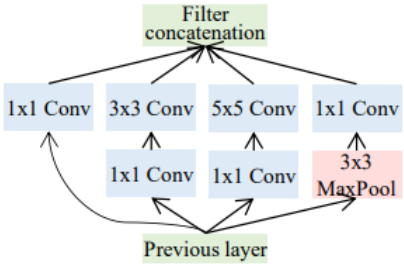


Figura 1.24: GoogleNet

Fuente: Deep Face Recognition: A Survey [25]

5. **FaceNet** [30]. Utiliza un amplio conjunto de datos para entrenar una GoogleNet, modificando la función de pérdida de la red.

6. **ResNet** [31]. Surgida en 2016, ResNet propone hacer que las capas aprendan un mapeo residual con referencia a las capas de entrada, en lugar de aprender directamente las conexiones de las capas subyacentes, con el objetivo de facilitar el entrenamiento de redes muy profundas (hasta 152 capas).

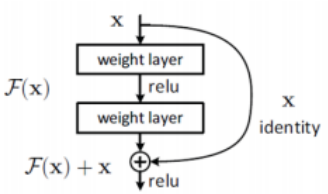


Figura 1.25: ResNet

Fuente: Deep Face Recognition: A Survey [25]

7. **SphereFace** [32]. Utiliza una ResNeT de 64 capas y propone una función de pérdida angular para aprender las características discriminativas de la imagen.

8. **SENet** [33]. Introduce, en 2017, un bloque 'Squeeze-and-Excitation' (SE) que recalibra de manera adaptativa las respuestas de funciones por canal.

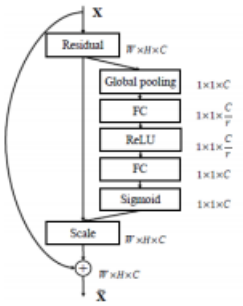


Figura 1.26: SENet

Fuente: Deep Face Recognition: A Survey [25]

Por último, el artículo previamente mencionado [25] también ofrece un estudio sobre la precisión de algoritmos basados en las arquitecturas descritas en esta sección:

Método	**Fecha**	**Arquitectura**	**Dataset**	**Accuracy**
DeepFace	2014	AlexNet	Facebook	97,35 %
FaceNet	2015	GoogleNet	Google	99,63 %
VGGface	2015	VGGNet	VGGface	98,95 %
Arcface	2018	ResNet	MS-Celeb-1M	99,83 %

Como se puede observar en la tabla, los porcentajes de acierto que muestran las técnicas de DL son bastante superiores a las que se mostraban con los algoritmos tradicionales de ML.

1.5. Conjuntos de datos para el entrenamiento y validación de técnicas de reconocimiento facial basadas en DL

Como se ha comentado en las secciones anteriores, los algoritmos de DL necesitan gran cantidad de datos para su entrenamiento. Uno de los principales problemas que existe actualmente con los conjuntos de datos es que, la mayoría de ellos, pertenecen a empresas privadas (Google, Facebook, etc.) y, en general, los conjuntos de datos públicos no son ni muy amplios ni muy numerosos.

No obstante, existen datasets públicos que son suficientemente amplios para poder trabajar con técnicas de DL y que permiten desarrollar pruebas de concepto sobre el reconocimiento facial. Stephen Balaban presenta una revisión sobre los datasets

más utilizados en el reconocimiento facial [34]. Dicha revisión, junto a la realizada por M.Wang y W. Deng [25], se resumen en la siguiente tabla:

Dataset	**Nº imágenes (MM)**	**Nº personas**	**Carácter**
Google Face Dataset	260	10.000.000	Privada
Megavii Face Classification (MFC)	5	20.000	Privada
Social Face Classification (SFC)	4,4	4.000	Privada
CASIA-WebFace	0,5	10.500	Pública
CASIA-WebFace	0,5	10.500	Pública
CelebFaces	0,2	10.100	Privada
Labeled Faces in the Wild (LFW)	0,01	5.800	Pública
MS-Celeb-1M	10	100.000	Pública
VGGFace2	3,3	9.100	Pública

El conjunto de datos que se utiliza en el presente trabajo es Labeled Faces in the Wild (LFW) [35].

1.6. Trazabilidad e integridad de la información en sistemas de identificación facial

Uno de los aspectos más importantes en cualquier sistema de identificación es garantizar la integridad y la trazabilidad de la información. Dado que, como se ha comentado previamente, los sistemas de identificación facial se podrían utilizar para registrar la jornada laboral en una empresa o para la gestión de accesos a entornos controlados, es muy importante que, cada vez que se identifique a un usuario, el sistema almacene un registro con la información básica y éste sea inmutable.

Realizando una revisión del estado actual sobre la implementación de estas dos características en los sistemas de identificación facial, no se han encontrado apenas artículos que traten esta temática. Por lo tanto, el sistema desarrollado que trata este documento mantendrá, como dos de sus características principales en el desarrollo del proyecto, la integridad y la trazabilidad de la información, ya que se considera algo novedoso y que apenas existe en el mercado.

Antes de analizar cómo se puede garantizar la trazabilidad y la integridad de la información en el sistema es importante entender bien ambos términos.

Trazabilidad

La trazabilidad de la información es un conjunto de procesos que permiten registrar e identificar las acciones realizadas sobre los datos. Es decir, sirve para determinar todos los pasos que recorre la información desde el inicio hasta el final. El objetivo es disponer de la información necesaria para poder auditar el sistema.

En el caso del reconocimiento facial, la trazabilidad se refiere de una forma más específica a ser capaz de almacenar la información necesaria para poder garantizar que todos los eventos de identificación producidos tienen su persistencia en el sistema. De esta manera, se podría determinar el número de veces que se ha identificado a una persona, así como la fecha y la hora en la que se ha realizado dicha identificación e, incluso, se podría llegar a almacenar la imagen de la persona identificada.

Integridad

En este caso, la integridad de la información se refiere a la condición de mantener los datos sin alteraciones desde el momento de su almacenamiento. Dicha integridad puede verse afectada por varios motivos: fallos en el sistema, ataques, etc. La integridad hace referencia tanto al proceso de modificar datos ya existentes como al proceso de borrado de los datos.

Aplicado al sistema de identificación facial, la integridad es, por tanto, un aspecto clave puesto que, cada vez que se almacene la traza de identificación en el sistema, ésta debe ser inmutable. La información persistida en el sistema no puede ser modificada ni borrada.

Implementación en un sistema

A la hora de garantizar la integridad y la trazabilidad de la información en un sistema existen varias alternativas.

El enfoque tradicional consistiría en tener una entidad certificadora que fuera la única capaz de acceder a la información y que se ocupara, a su vez, de garantizar la trazabilidad de la información. Este enfoque corre el riesgo de que si la entidad certificadora es atacada, el atacante podría acceder a la información y modificarla o borrarla. Además, en caso de fallo del sistema, si la entidad certificadora cae, el sistema dejaría de funcionar.

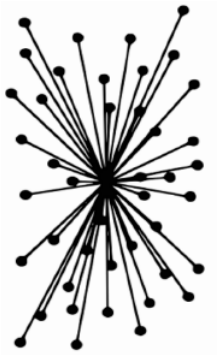


Figura 1.27: Arquitectura centralizada

Fuente: Desarrollo de un sistema de trazabilidad en entornos IoT mediante Hyperledger [36]

Otra posible solución, que es la más utilizada actualmente, sería emplear una arquitectura descentralizada en la que hubiera varias entidades certificadoras que tuvieran los mismos roles que en el enfoque anterior. Esto garantiza que, ante el fallo de una entidad en el sistema, habría otras de soporte. Sin embargo, en el caso de que una de las entidades fuera atacada y su información fuera modificada, se pondría en peligro la integridad de la información, dado que se requeriría una alta interoperabilidad entre el resto de entidades certificadoras y es un proceso muy costoso.

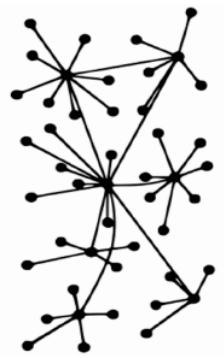


Figura 1.28: Arquitectura descentralizada

Fuente: Desarrollo de un sistema de trazabilidad en entornos IoT mediante Hyperledger [36]

Por último, existe una arquitectura alternativa a la que, probablemente, tiendan muchas de las actuales. Se trata de las arquitecturas distribuidas. Bajo este enfoque, no existen entidades centrales, sino que cada uno de los nodos actúa como entidad «central» y está conectado con el resto de la red. Esto garantiza que el sistema sea tolerante a fallos, pues hay una red por debajo que soporta el sistema, y que se mantenga la integridad de la información ya que, para modificar la información, ésta debe ser modificada en un número suficientemente grande de nodos.

La tecnología Blockchain permite implementar este tipo de arquitecturas, por lo que se abordará en detalle en la siguiente sección; es la solución que adopta el sistema de identificación desarrollado con objeto de que las evidencias de identificación sean almacenadas con preservación de su integridad y evitando fallos de punto único (Single Point of Failure -SPOF-).

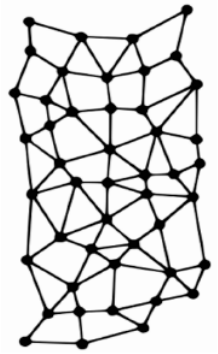


Figura 1.29: Arquitectura distribuida

Fuente: Desarrollo de un sistema de trazabilidad en entornos IoT mediante Hyperledger [36]

1.7. Blockchain

El concepto Blockchain (BC) o cadena de bloques es muy amplio y no hay definición exacta actualmente. No obstante, se pueden establecer una serie de características comunes. BC se refiere a un tipo de tecnología de registro distribuido o Distributed Ledger Technology (DLT) [37], también conocida como libro de registros o ledger. Este tipo de tecnología se hace popular debido, en gran parte, al surgimiento de Bitcoin [38].

DLT, por su parte, es una estructura de datos distribuida que es compartida y replicada por todos los nodos que pertenezcan a la red. El libro de registros contiene todas las transacciones registradas en la red indicando: procedencia, destinatario y contenido. Estas transacciones son agrupadas en bloques y son estos bloques los que, finalmente, se almacenan en la cadena junto con información adicional para enlazar un bloque con otro. Su objetivo es poder recuperar la información y garantizar su integridad. La integridad se consigue ya que, una posible modificación en el contenido de una transacción, requiere la alteración de cada uno de los bloques de la cadena para ocultar dicha modificación, tarea prácticamente imposible. En el momento en el que una transacción se registra en la cadena de bloques se considera como válida. La validación de transacciones es esencial para proporcionar a la red consistencia y prevenirla de alteraciones indeseadas. Para ello, cada red debe establecer ciertas

reglas con el objetivo de determinar si una nueva transacción es válida o no. Esta validación suele estar asociada a la firma de la transacción.

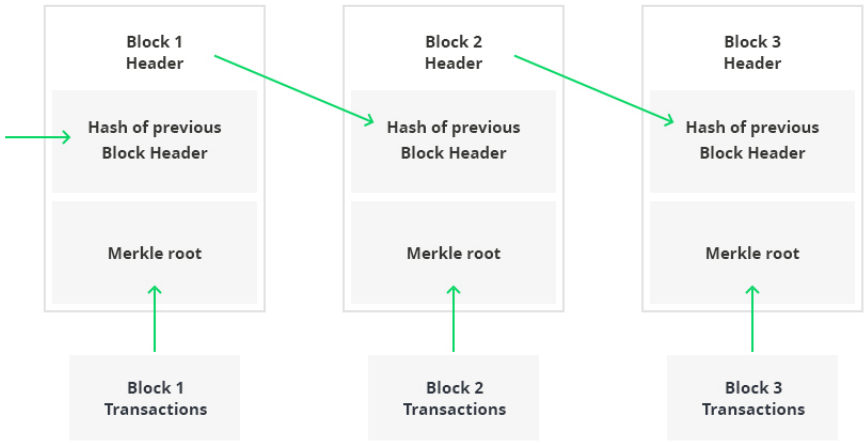


Figura 1.30: Estructura de la cadena de bloques

Fuente: Desarrollo de un sistema de trazabilidad en entornos IoT mediante Hyperledger [36]

Cada uno de los bloques de la cadena se identifica por un valor hash, obtenido a partir de la aplicación por partida doble de una función hash a la cabecera del bloque. Adicionalmente, los bloques contienen: cabecera del bloque (con metadatos sobre dicho bloque) y un árbol de Merkel (sintetiza todas las transacciones de un bloque).

Las funciones hash tienen una serie de características que permiten garantizar, entre otras cosas, la integridad de la información:

- **Determinismo**. Para un conjunto de datos dado siempre se obtiene el mismo valor hash.
- **Eficiencia**. Son computacionalmente eficientes.
- **No reversibilidad**. No existe una forma directa de conseguir obtener los datos originales a partir del valor hash.
- **Sensibilidad a cambios**. Un pequeño cambio en la entrada produce multitud de cambios en la salida.
- **Resistencia a colisiones**. La probabilidad de obtener el mismo valor hash ante un mismo conjunto de datos es prácticamente nula.

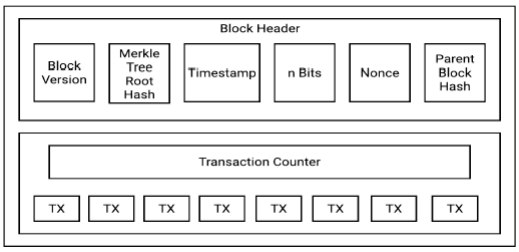


Figura 1.31: Estructura de un bloque

Fuente: Desarrollo de un sistema de trazabilidad en entornos IoT mediante Hyperledger [36]

Las principales características de la tecnología Blockchain son:

- **Arquitectura distribuida**. Presenta un ledger distribuida en la que cada nodo tiene una copia de toda la información de la red.
- **Confianza**. La confianza de la red reside en el consenso alcanzando entre los nodos, sin que exista una entidad central que controle y gobierne la red.
- **Transparencia**. Las transacciones son transparentes, verificables y trazables.
- **Integridad**. Una vez se ha introducido la información, ésta es inmutable y no puede ser modificada por nadie.
- **Bases criptográficas**. Blockchain se basa en las propiedades y garantías de seguridad que ofrecen las funciones hash y en la firma digital para proteger criptográficamente la información.

Problema de los generales bizantinos

El problema de los generales bizantinos [39] es un juego de lógica creado para ilustrar el dilema de lograr un consenso entre un conjunto de entidades con un objetivo común cuando, entre ellas, pueden existir traidores (entidades con objetivos opuestos que intenten dinamitar el proceso) [40]. En el planteamiento del problema se supone que las comunicaciones entre las entidades son limitadas e inseguras.

Dicho problema se representa de manera análoga en un escenario de guerra, donde un grupo de generales bizantinos se encuentran acampados y preparados para atacar al enemigo. Tras observar al enemigo, los generales deben acordar un plan común que les permita vencerlo. Para ello, los generales se comunican únicamente a través de mensajeros, con la posibilidad de que algún general sea un traidor y, por ende, envíe mensajes falsos. Para solucionar el problema, se debe asegurar que todos los generales

leales acuerden el mismo plan, evitando que los traidores consigan modificarlo. Es decir, deben llegar a un consenso entre ellos.

La tecnología Blockchain permite alcanzar el consenso en la red a través de la implementación de protocolos de consenso distribuido.

Consenso

El consenso es un factor clave en las cadenas de bloques, ya que es donde reside la confianza. Dentro de BC, las firmas digitales se utilizan para validar que una transacción es firmada por quién dice ser. No obstante, existe el problema de enviar más de una vez la misma transacción, donde la firma es válida (problema de doble gasto) [41]. Además, el problema de los generales bizantinos detallado previamente también debe ser resuelto mediante el consenso, para evitar que un conjunto de nodos se comporten de forma maliciosa.

Para evitar estos problemas, la red debe tener ciertas reglas y principios comunes para todos los nodos, así como reglas de validación de transacciones. El protocolo de consenso distribuido es empleado para conseguir la validación de las transacciones. Existen varios algoritmos para el protocolo de consenso, siendo los más utilizados los siguientes:

- **Proof of Work (PoW)** [42]. Protocolo donde se utiliza la capacidad de cómputo de hardware para resolver cálculos matemáticos y determinar quién es el creador de un nuevo bloque. El creador recibe una recompensa, por ejemplo, en forma de criptomoneda.
- **Proof of Stake (PoS)** [43]. Protocolo basado en elegir de manera determinista al creador de un nuevo bloque, en base a su riqueza o participación. La recompensa recibida por el creador es una comisión pagada por cada transacción.

Tipos de cadena de bloques

Existen una serie de características que permiten categorizar las redes Blockchain, para poder evaluarlas y compararlas a la hora de implementar sistemas basados en esta arquitectura. Una posible categorización sería la siguiente [44]:

- **Accesibilidad**: pública (cualquiera puede acceder a la red), permisionada (control de acceso) o híbrida.

- **Permisibilidad**: en las cadenas de bloques privadas pueden existir roles para asignar diferentes privilegios dentro de la red. Estos roles también pueden afectar directamente a la participación en el protocolo de consenso.
- **Descentralización**: las cadenas de bloques públicas son totalmente descentralizadas, mientras que las privadas y las híbridas son parcialmente centralizadas.
- **Tipología de transacciones**: modelo Unspent Transaction Output (UTXO) al estilo Bitcoin [38] o modelo cuenta-balance al estilo Ethereum [45].

Smart Contracts

Los Smart Contracts o contratos inteligentes hacen referencia a un acuerdo entre dos o más partes en el que se define lo que se puede hacer, cómo se puede hacer, etc. siendo capaz, además, de ejecutarse de manera automática y sin intermediarios [46].

Aplicando el concepto a la cadena de bloques, esto se refiere a programas informáticos que implementan una serie de acuerdos previamente definidos y establecidos entre dos o más partes. Estos programas se ejecutan automáticamente como resultado de que se cumplan determinadas condiciones. Este tipo de contratos permiten ampliar las funcionalidades distribuidas de una cadena de bloques como, por ejemplo, realizar transacciones financieras entre dos o más participantes.

Como cualquier otro tipo de software, los Smart Contracts están sujetos a problemas de seguridad. De hecho, algunos autores los han categorizado en base a sus problemas de seguridad [47]:

- **Contratos pródigos**: en este tipo de contratos se consigue que las transacciones lleguen a un destinatario no deseado.
- **Contratos suicidas**: contratos que se cierran al activarse un requisito de salida por parte de quien realiza el ataque.
- **Contratos codiciosos**: en este tipo de contratos el usuario deja de tener legitimidad sobre sus criptodivisas. Puede deberse a una mala práctica o a una redacción equivocada.

Implementaciones y consorcios de Blockchain

Existen multitud de implementaciones diferentes de Blockchain y, cada vez, surgen más. En esta sección se realiza una revisión general de las implementaciones

más utilizadas actualmente. Según «Deloitte's 2019 Global Blockchain Survey» [48], en 2019, las cadenas de bloques privadas o permisionadas ya han superado a las públicas.

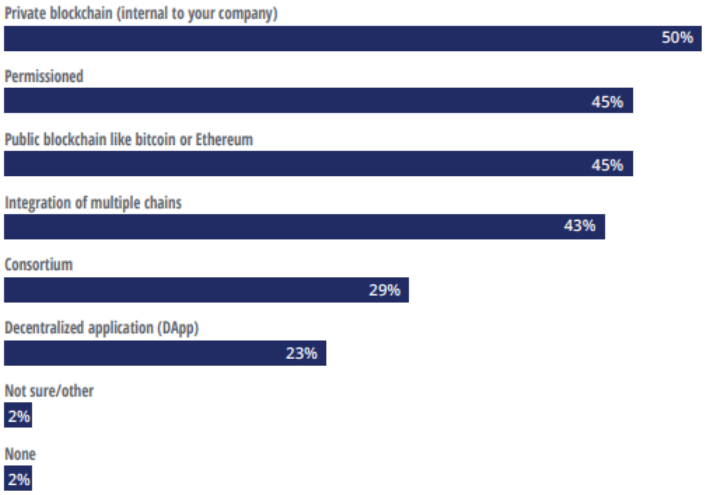


Figura 1.32: Porcentaje de uso de los modelos de Blockchain

Fuente: Deloitte [48]

A continuación, se detallan las cadenas de bloques más utilizadas en el mercado:

- **Bitcoin**. Se podría decir que el paradigma de Blockchain surge con Bitcoin y Satoshi Nakamoto [38]. Bitcoin se define como «una red consensuada que permite un nuevo sistema de pago y una moneda completamente digital». El término Bitcoin se puede referir tanto a la criptodivisa (BTC) como al protocolo de código abierto en el que se basa la tecnología. Es una cadena de bloques pública.
- **Ethereum** [45]. Es una plataforma para la creación de aplicaciones descentralizadas y acuerdos denominados Smart Contracts. Incluye un lenguaje de programación, un protocolo y una criptodivisa (ETH). Es una cadena de bloques pública.
- **Litecoin** [49]. Es una moneda criptográfica desarrollada de forma independiente a partir de la base de Bitcoin. Surgió como mejora frente a Bitcoin, con el objetivo principal de ejecutar las transacciones de manera más rápida y de optimizar el proceso de minado.
- **Dash** [50]. Es un sistema de pago descentralizado cuyo objetivo principal es anonimizar las transacciones.

En el informe de Deloitte citado anteriormente [48], se indica que existe una tendencia a la alza durante los últimos años por la que las empresas e instituciones interesadas en investigar acerca de Blockchain lo hacen de manera conjunta a través de consorcios. El hecho de crear consorcios entre empresas, bancos, startups, etc. favorece la implementación y comercialización de Blockchain.

Actualmente, existen más de cien consorcios y el número sigue creciendo de manera global. Entre los consorcios más conocidos se encuentran:

- **R3** [51]: es una empresa que se dedica al desarrollo de la tecnología de registro distribuido. R3 domina el consorcio en el que participan más de setenta empresas mundiales, entre las que se encuentran grandes corporaciones financieras (BBVA, Barclays, Goldman Sachs, etc.). Todas ellas han realizado proyectos interesantes, como la demostración de que Blockchain permite realizar transacciones bancarias más rápidas y seguras que de la manera tradicional. No obstante, en los últimos años, se han producido salidas de entidades de relevancia como Goldman Sachs o J.P. Morgan Chase, dado que no se ha podido proveer ninguna implementación comercial de la tecnología.
- **Hyperledger** [52]: iniciativa que incluye varios proyectos de código abierto iniciado por la Fundación Linux que tiene como objetivo avanzar en la tecnología Blockchain en diferentes sectores. Este caso se analizará en detalle en el siguiente apartado.
- **Ethereum Enterprise Alliance (EEA)** [53]: consorcio fundado por un grupo de startups, de empresas innovadoras y de empresas multinacionales con el objetivo de desarrollar, apoyar y promocionar nuevos proyectos Blockchain basados en Ethereum. Pertenecen a este consorcio empresas como: Intel, Microsoft, J.P. Morgan, Banco Santander, etc. Dicho consorcio todavía no ha lanzado propuestas concretas de utilización sobre esta tecnología.
- **The Post Trade Distributed Ledger Group** [54]: constituido por la Bolsa de Londres, incluye aproximadamente cuarenta entidades financieras para trabajar sobre tecnología de registro distribuido.

Hyperledger

Hyperleder [52] es un consorcio colaborativo de código abierto que se creó en 2015 por la Fundación Linux con el objetivo de investigar acerca de la tecnología de registro distribuido. En concreto, se trabaja sobre la Blockchain de carácter privado y orientada al ámbito empresarial, donde ve potencial adaptación en los procesos

actuales para su mejora. Además, trata de establecer estándares para garantizar la interoperabilidad y estabilidad de los proyectos y evitar la fragmentación entre ecosistemas.

En el consorcio participan empresas de todo tipo, desde compañías tecnológicas (IBM o Intel), compañías dedicadas a Blockchain (R3) hasta entidades financieras (BBVA o J.P. Morgan).

Dentro de los proyectos más importantes desarrollados por el consorcio se encuentran:

- **Hyperledger Fabric (HF)**. HF es el proyecto más conocido dentro del consorcio Hyperledger. Está orientado al desarrollo de soluciones que implementen la tecnología Blockchain dentro del ámbito empresarial. Ofrece características muy interesantes como ausencia de criptodivisa, escalabilidad a través de una arquitectura modular, alto rendimiento, privacidad e identidad o algoritmos de consenso configurables.

  La principal característica es que tiene carácter privado y permisionado, por lo que permite restringir y controlar el acceso a la red y a sus activos.

  El ledger de HF está formado por dos componentes: estado del mundo, que describe el estado del libro de registros en un momento temporal, y el log de transacciones, que almacena todas las transacciones del mundo.

  La información almacenada en el ledger proviene de la ejecución de la lógica del sistema por parte de soluciones externas. Dichas soluciones externas se comunican con el ledger a través de chaincodes o Smart Contracts. Los Smart Contracts definen las reglas de las transacciones y están implementados en Go o en Java.

- **Hyperledger Composer (HC)**. HC es una herramienta de código abierto para desarrollar redes de negocio basadas en blockchain que estén construidas sobre la infraestructura de Hyperledger Fabric. Su objetivo principal es facilitar el proceso de definición de la red de negocio y del desarrollo de los Smart Contracts.

- **Hyperledger Indy**. Propone una solución para la identificación descentralizada basada en el protocolo criptográfico de conocimiento cero o Zero Knowledge Proof, el cual trata de reducir la cantidad de información personal en los procesos de identidad digital. Además, dispone de Self Sovereignty Identity (SSI).

- **Hyperledger Burrow**. Blockchain privada basada en Ethereum.
- **Hyperledger Iroha**. Tiene como objetivo dotar a los dispositivos móviles de una plataforma Blockchain simple y modular que permita el despliegue de Smart Contracts.

Capítulo 2

Gestión de proyecto software

En este capítulo se detalla la gestión del proyecto sobre el sistema de identificación facial desarrollado. Dicha simulación incluye la implantación del sistema en un entorno productivo empresarial.

2.1. Alcance del proyecto

El proyecto engloba la puesta en producción de un sistema de identificación basado en técnicas de reconocimiento facial bajo una infraestructura Blockchain e incluye:

- Instalación de cámaras web IP para la captura de imágenes.
- Implementación de un modelo de identificación facial desarrollado mediante técnicas de Deep Learning.
- Creación de una infraestructura Blockchain para la persistencia de la información.

2.1.1. Definición del proyecto

El objetivo de este proyecto es desarrollar un sistema de identificación basado en técnicas de reconocimiento facial. Dicho sistema se apoya en la aplicación de tecnologías Deep Learning, para el uso del modelo de identificación facial, y de Hyperledger Fabric, para la persistencia del sistema. Así, el sistema capturará imágenes de diferentes individuos y permitirá identificarlos en él.

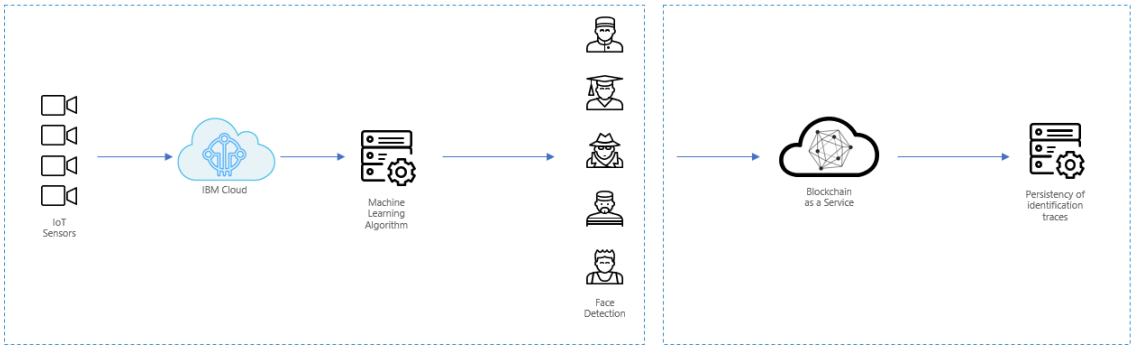


Figura 2.1: Esquema del proyecto

Fuente: Elaboración propia

2.1.2. Estimación de tareas y recursos

Las tareas principales que se han identificado para implementar el producto son:

- Preparación del entorno Cloud.
- Creación del conjunto de datos con los individuos que formarán parte del sistema.
- Implementación del modelo de identificación facial.
- Pruebas unitarias del modelo de identificación.
- Definición de la arquitectura Blockchain.
- Desarrollo de la lógica de negocio de la red Blockhain.
- Despliegue del modelo de red Blockchain.
- Pruebas unitarias de la arquitectura Blockchain.
- Persistencia de los registros de identificación en la red Blockchain.
- Pruebas globales del sistema.
- Despliegue en producción del sistema.

Respecto a los recursos necesarios para el desarrollo del proyecto, se diferenciará entre recursos humanos y recursos materiales:

- Recursos humanos. Se requiere un científico de datos, un ingeniero DevOps, un ingeniero Blockchain, un ingeniero de pruebas y un jefe de proyectos. En este caso, todos estos papeles han sido realizados por el autor.

- Recursos materiales. Se desarrollará el sistema en un entorno Cloud, el cual proporciona Infrastructure as a Service (IaaS). Únicamente sería necesario la adquisición de dos cámaras webs IP.

2.1.3. Presupuesto

A continuación, se detalla un presupuesto estimado para el coste total de este proyecto.

Coste de personal

Tabla 2.1: Presupuesto de personal

Tarea	Perfil	Horas	€/Hora	Total
Preparación del entorno Cloud	Ing. DevOps	16	60	960 €
Generación del dataset	Científico datos	32	100	3200 €
Implementación modelo identificación facial	Científico datos	80	100	8000 €
Pruebas del modelo de identificación	Ing. Pruebas	16	80	1280 €
Definición arquitectura Blockchain	Ing. Blockchain	40	100	4000 €
Lógica de negocio de Blockchain	Ing. Blockchain	40	100	4000 €
Despliegue red Blockchain	Ing. Blockchain	8	100	800 €
Pruebas de Blockchain	Ing. Pruebas	16	80	1280 €
Registro trazas identificación en Blockchain	Ing. Blockchain	16	100	1600 €
Pruebas globales del sistema	Ing. Pruebas	16	80	1280 €
Despliegue en producción del sistema	Ing. DevOps	16	60	960 €
Supervisión del Proyecto	Jefe de Proyecto	20	120	2400 €
Total				29760 €

Coste del hardware

Se propone el uso de IBM Cloud, ya que permite el despliegue de Hyperledger Fabric en la nube. Para estimar los costes, se ha utilizado la propia calculadora facilitada por IBM Cloud [55]. La estimación realizada se hará para estimar el coste anual de la plataforma, dado que, en este tipo de infraestructuras, se paga por uso de manera mensual.

Para realizar la estimación de la plataforma Blockchain de IBM Cloud es necesario determinar el número de horas de uso mensuales del sistema. Se propone un uso de

40 horas semanales, equivalentes a una jornada laboral estándar. Extrapolando las horas, sería un total de 160 horas mensuales, que equivalen a un precio anual de 556,80 €.

Por otro lado, para la integración del modelo Deep Learning de identificación facial, se requiere procesamiento GPU y gran potencia de cómputo. Se propone el uso de una arquitectura con una GPU Nvidia P100 y 128GB RAM. La elección de esta arquitectura se factura a 1,6895 € por hora de uso en cada nodo. Suponiendo una arquitectura de un nodo y un período anual, se obtendría un coste de, aproximadamente, 1087,4 €.

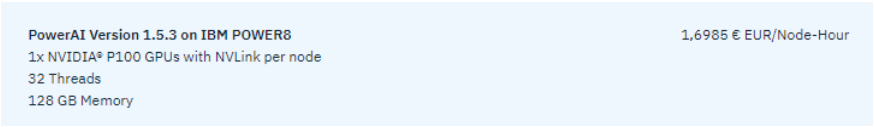


Figura 2.2: Precio infraestructura IBM Cloud

Fuente: IBM Cloud

Por último, para la instalación de las cámaras web se utilizará el servicio IoT de IBM Cloud que ofrece un plan gratuito. En caso de que se requiera mayor capacidad, se ampliarían los requerimientos.

Por lo tanto, el coste hardware total anual estimado sería de 1645 €.

Coste total

Sumando los costes de recursos humanos a los recursos físicos, se obtendría un importe total anual de, aproximadamente, 31405 €.

Tabla 2.2: Presupuesto total

Concepto	Coste (Euros)
Costes de personal	29760
Costes de hardware	1645
Subtotal	31405
IVA (21 %)	6595.05
Total Proyecto	**38000,05 €**

2.2. Plan de trabajo

En esta sección se detalla el plan de trabajo estimado que se ha desarrollado para hacer frente a la implementación del sistema de identificación basado en técnicas de reconocimiento facial.

2.2.1. Identificación de tareas

Las tareas identificadas en el proyecto son las que se exponen en la sección 2.1.2. No obstante, en esta sección se expone con mayor detalle:

- Preparación del entorno Cloud. Se debe preparar y configurar el entorno de IBM Cloud para poder empezar con el desarrollo. Esta tarea conlleva el despliegue de todos los servicios necesarios.
- Creación del dataset con los individuos que formarán parte del sistema. Es necesario generar un dataset con todas las personas que participarán en el sistema para que el modelo sea capaz de identificarlos.
- Implementación del modelo de identificación facial. Configuración, adaptación e implementación del modelo de detección facial.
- Pruebas unitarias del modelo de identificación.
- Definición arquitectura Blockchain. Se debe diseñar la red de Blockchain con todos los componentes que la conforman.
- Desarrollo de la lógica de negocio de la red Blockhain. Configuración de las reglas de la red y de la implementación de las funciones necesarias para su uso.
- Despliegue del modelo de red Blockchain. Implementación de Hyperledger Fabric.
- Pruebas unitarias de la arquitectura Blockchain.
- Persistencia de los registros de identificación en la red Blockchain. Cada vez que se identifique un individuo, el sistema almacenará un registro en la cadena de bloques con la información esencial.
- Pruebas globales del sistema.
- Despliegue en producción del sistema.

2.2.2. Estimación de tareas

Las tareas se han estimado en horas y el detalle de las mismas se encuentra en la tabla 2.1.

2.2.3. Planificación de tareas

A la hora de planificar las tareas, se ha tratado de dividir la solución en dos grandes bloques, de tal manera que las tareas se puedan paralelizar en la medida de lo posible y no sean totalmente dependientes unas de otras. Dicha planificación se indica en el siguiente diagrama y, como se puede observar, tiene una duración total de, aproximadamente, cinco semanas.

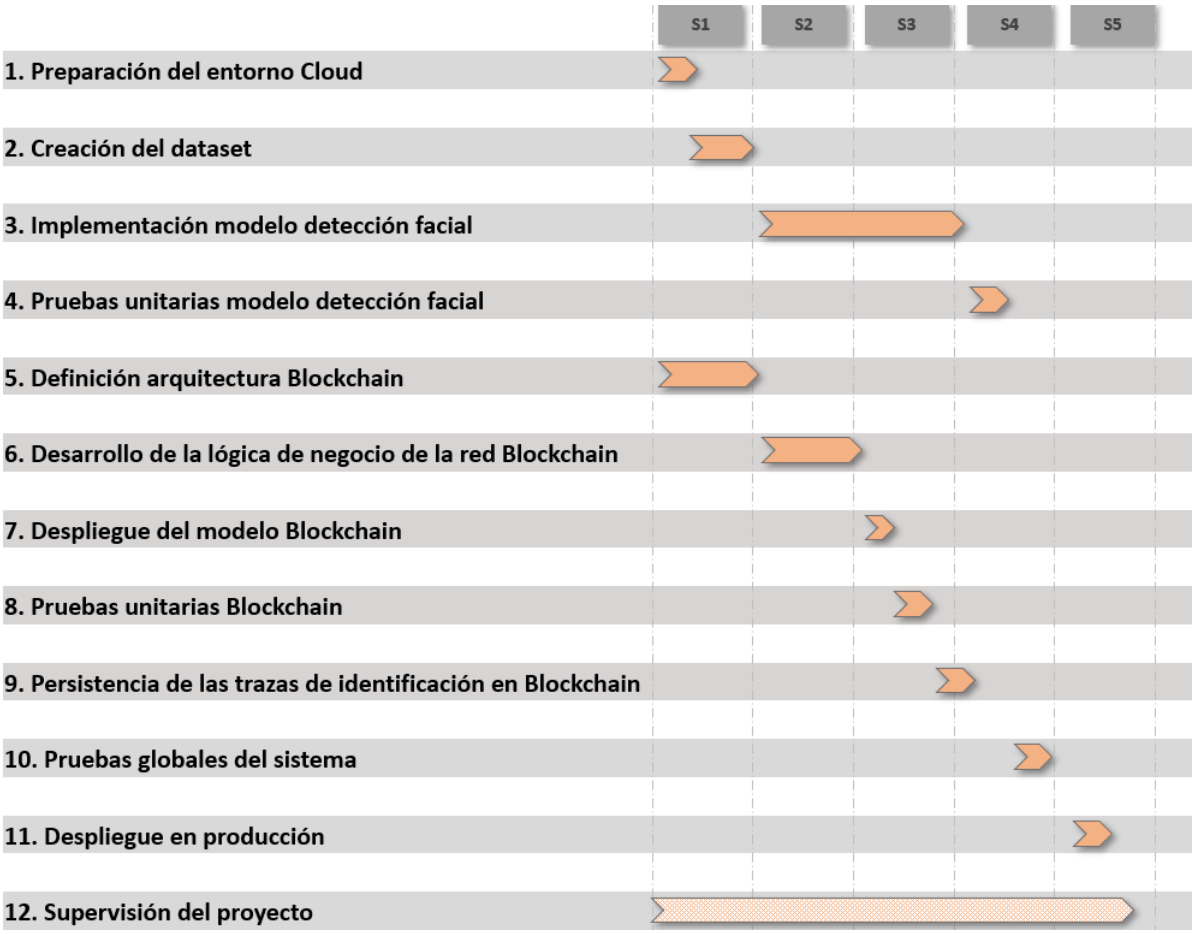


Figura 2.3: Planificacion de tareas

Fuente: Elaboración propia

2.3. Gestión de recursos

En esta sección se detallan los recursos humanos necesarios para hacer frente a la implementación del sistema de identificación basado en técnicas de reconocimiento facial.

2.3.1. Especificación de recursos

Los recursos humanos necesarios para desarrollar el proyecto se encuentran listados en la seccion 2.1.2. No obstante, en esta sección se ofrece un mayor detalle sobre los mismos:

- Científico de datos. Especialista en técnicas de Deep Learning y analítica de datos.
- Ingeniero DevOps. Recurso con conocimientos, tanto de desarrollo como de operaciones.
- Ingeniero Blockchain. Especialista en infraestructuras basadas en cadenas de bloques.
- Ingeniero de pruebas. Debe ser capaz de establecer un plan de pruebas completo que garantice la robustez del sistema.
- Jefe de proyectos. Persona con experiencia en la implantación de proyectos de temática similar.

2.3.2. Asignación de recursos

La asignación de los recursos a las tareas aparece detallado en la Tabla 2.1.

2.4. Gestión de riesgos

En esta sección se detallan los posibles riesgos que pueden surgir durante el desarrollo del sistema y su evaluación de impacto.

2.4.1. Identificación de riesgos

Para identificar los riesgos, se ha tomado como base la documentación del proyecto y se ha realizado una revisión del estado del arte de otros proyectos similares. A partir de ahí, se han identificado los siguientes riesgos técnicos:

- Caída del entorno Cloud.
- Problemas de identificación del modelo de detección facial.
- Error en la captura de las imágenes.
- Fallos en la red de escritura Blockchain.

Además, se debe tener en cuenta que también existen riesgos derivados del propio desarrollo del proyecto como son:

- Desviaciones en la planificación realizada.
- Problemas con los recursos humanos del proyecto.
- Sobrejauste del presupuesto.

2.4.2. Análisis de riesgos

Para analizar los riesgos, se ha realizado un estudio cualitativo de los mismos en base a la guía del PMBOK [56]. Bajo estos criterios, se utiliza una tabla de probabilidad con cinco niveles que permite ubicar el riesgo en un nivel de ocurrencia:

Tabla 2.3: Niveles de ocurrencia

Nivel	Descriptor	Descripción
1	Raro	El evento puede ocurrir por circunstancias expecionales
2	Improbable	El evento es difícil que ocurra en algún momento
3	Posible	El evento puede ocurrir en algún momento
4	Probable	El evento ocurre en muchas ocasiones
5	Casi seguro	Se espera que el evento ocurra casi siempre

Además, se debe medir el nivel de impacto provocado por los riesgos si éstos llegan a materializarse. Para ello, se utiliza la siguiente tabla:

Tabla 2.4: Niveles de impacto

Nivel	Descriptor	Descripción
1	Insignificante	Consecuencias o efectos mínimos
2	Menor	Bajo impacto
3	Moderado	Impacto moderado
4	Mayor	El efecto sería notable
5	Catastrófico	Consecuencias desastrosas

Para evaluar los riesgos identificados en la sección anterior, se realiza una matriz de probabilidad e impacto en la que se refleja la multiplicación del nivel de ocurrencia por el nivel de impacto, siendo éste el resultado que determina el riesgo:

PROBABILIDAD	IMPACTO				
	Insignificante (1)	Menor (2)	Moderado (3)	Mayor (4)	Catastrófico (5)
Raro (1)					
Improbable (2)					
Posible (3)					
Probable (4)					
Casi seguro (5)					

Figura 2.4: Matriz de impacto de riesgos

Fuente: `https://ciberconta.unizar.es/ifinanzas/08-P-matrizriesgo.htm`

Aplicando dicha matriz sobre el proyecto actual, se obtiene la siguiente tabla:

Tabla 2.5: Determinación del riesgo del proyecto

Riesgo	Nivel de riesgo
Caída del entorno Cloud	Raro pero catastrófico: 5
Problemas de identificación del modelo de detección facial	Posible y moderado: 9
Error en la captura de las imágenes	Posible y moderado: 9
Fallos en la red de escritura Blockchain	Improbable y menor: 4
Desviaciones en la planificación realizada	Posible y menor: 6
Problemas con los recursos humanos del proyecto	Posible y mayor: 12
Sobrejauste del presupuesto	Posible y moderado: 9

Capítulo 3

Desarrollo de la solución

Tras la recopilación de información obtenida durante la etapa de análisis y en la revisión del estado del arte, es importante realizar un diseño del proyecto y dotarlo de una arquitectura que garantice que se cumplen todos los requerimientos objetivo del proyecto. Este capítulo ayudará al lector a comprender el funcionamiento del sistema de identificación basado en reconocimiento facial implementado.

3.1. Descripción de la solución adoptada

El objetivo del proyecto es desarrollar una prueba de concepto sobre un sistema de identificación basado en técnicas de reconocimiento facial. Dicho sistema se ha desarrollado mediante el uso de tecnologías Deep Learning, para el uso del modelo de identificación facial, y de Hyperledger Fabric, para la persistencia del sistema.

El uso de tecnologías Deep Learning permite simular el comportamiento humano e, incluso, mejorarlo en la resolución de determinados problemas. Además, una de las ventajas que ofrece su uso es que permite el tratamiento de datos masivos, lo cual ayuda, en gran medida, a optimizar la tasa de acierto en el problema de la detección facial.

Por otro lado, la elección de Blockchain, a través de su implementación en Hyperledger Fabric, se debe, principalmente, a que permite garantizar la trazabilidad de la información en el sistema. Si bien es cierto que dicha trazabilidad se puede realizar utilizando metodologías tradicionales, es evidente que estos sistemas carecen de seguridad suficiente para garantizar que la información ha permanecido inmutable desde su registro.

De cara a facilitar la comprensión del sistema, a continuación se expone un diagrama general de la solución propuesta:

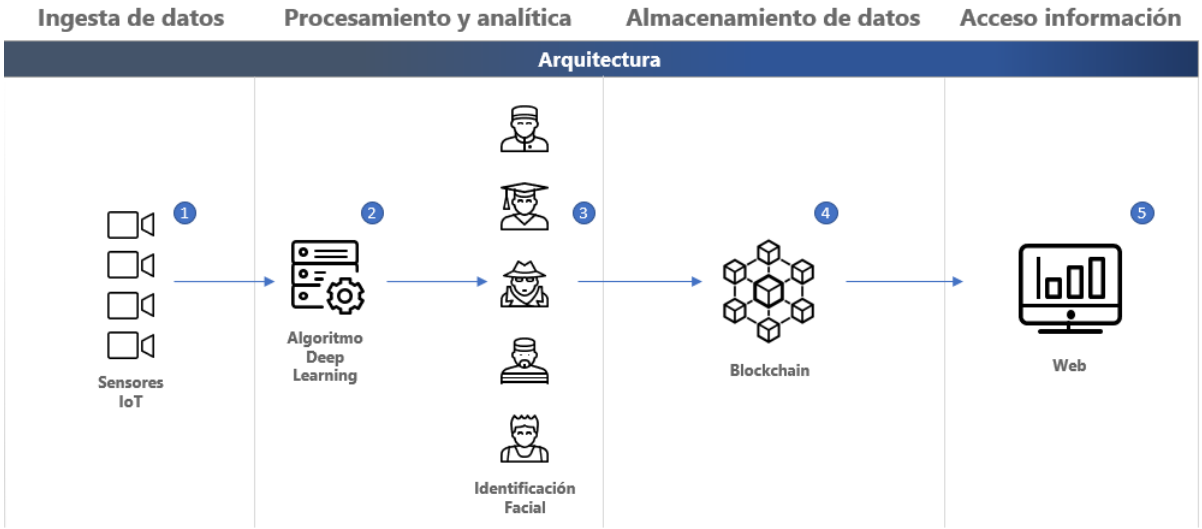


Figura 3.1: Arquitectura de la solución propuesta

Fuente: Elaboración propia

Dicho diagrama representa:

1. Mediante una cámara web se captura la imagen en tiempo real de una persona.
2. Se aplica un modelo de detección facial mediante técnicas de Deep Learning sobre la imagen capturada.
3. Con el modelo se consigue reconocer a la persona e identificarla.
4. Tras reconocer a la persona, el registro de la identificación se almacena en una arquitectura basada en cadena de bloques.
5. Por último, se pueden consultar los datos a través de una interfaz web.

3.1.1. Diseño y arquitectura del sistema

El objetivo de esta sección es detallar el diagrama de la solución mostrado anteriormente presentando el diseño elegido y la arquitectura desarrollada, justificando el porqué de ambos.

A la hora de diseñar la solución, se ha pensado en implementar una prueba de concepto funcional y sencilla que permita comprender la lógica del sistema y que pueda servir como base para el desarrollo de un sistema completo. Para ello, durante la realización del proyecto, se ha tratado de seguir, en la medida de lo posible, el

patrón de diseño software Modelo-Vista-Controlador (MVC) [57]. No obstante, no se ha realizado el desarrollo de ninguna interfaz de usuario ad-hoc en el proyecto, por lo que la capa de la vista no estaría implementada explícitamente.

Bajo este paradigma, las interacciones del usuario, la lógica de negocio y el resto de elementos del sistema son separados y manejados por tres tipos de objetos distintos:

- **Modelo**: contiene una representación de la información que maneja el sistema, la lógica de negocio y los mecanismos de persistencia.
- **Vista**: contiene los mecanismos de interacción de la parte visual con el sistema.
- **Controlador**: actúa como intermediario del modelo y de la vista.

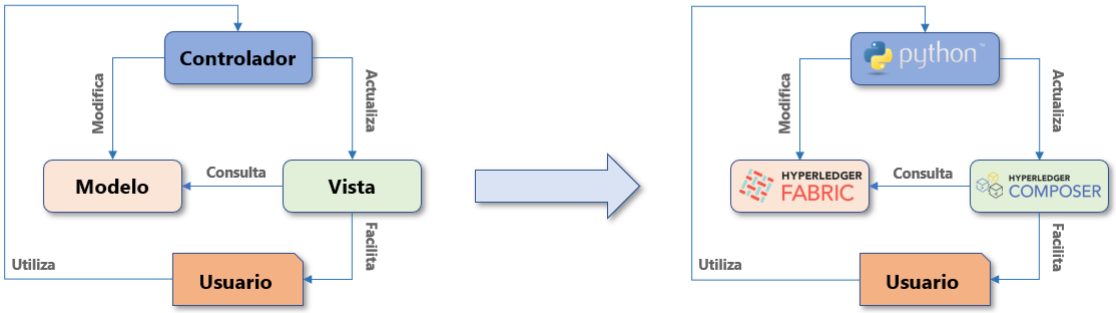


Figura 3.2: Modelo-Vista-Controlador aplicado

Fuente: Elaboración propia

La figura anterior muestra el esquema clásico del patrón Modelo-Vista-Controlador y su traducción en el sistema desarrollado. Como se puede observar, no existe una capa de vista desarrollada, sino que se utiliza directamente Hyperledger Composer para la visualización de los datos, además de poder consumirse a través de una API-REST.

La arquitectura se presenta con dos entornos diferenciados con distintos servicios y plataformas. Por un lado, se engloba en un entorno Cloud, en el cual se realizan los procesos relativos a la captura de imágenes y a la aplicación del modelo de detección facial y, por otro, en un entorno local en el que se ha desplegado Hyperledger Fabric como tecnología para implementar la cadena de bloques.

El uso de un entorno Cloud se debe principalmente a que, para realizar el procesamiento de grandes volúmenes de datos y para aplicar modelos basados en Deep Learning, se necesitan componentes hardware muy potentes de los que no se dispone

y que requerirían un desembolso económico inicial notable. Los entornos Cloud permiten, además, escalar la solución de manera óptima y realizar el despliegue de la infraestructura de manera muy ágil, lo que posibilita la implementación de pruebas de concepto como la tratada en este documento. Es por ello que se ha escogido utilizar la plataforma Google Colaboratory que ofrece, de manera gratuita, un entorno GPU en el backend de Google Compute Engine que permite aplicar modelos de Deep Learning basados en redes convolucionales de manera eficiente. Además, Google Colaboratory ofrece la posibilidad de ejecutar notebooks de Python, los cuales se han utilizado para desarrollar el proyecto.

Por último, el entorno local se ha utilizado para el despliegue de la capa de persistencia, siguiendo una arquitectura de cadena de bloques. El sistema mantiene su persistencia en una cadena de bloques de carácter privada y permisionada, como es Hyperledger, lo cual incrementa la seguridad frente a las cadenas de bloques públicas, ya que se puede controlar quién puede acceder y bajo qué privilegios a la red, proporcionándole, así, mayor confidencialidad. Es, además, una gran ventaja para el ámbito empresarial, dado que pueden desplegar su arquitectura Blockchain sin necesidad de que ésta sea expuesta públicamente a cualquier usuario ajeno a la organización. Por tanto, la ventaja principal de la elección de una arquitectura basada en esta tecnología es garantizar la integridad de la información.

La siguiente figura presenta de forma más exhaustiva la disposición de los elementos, concretando las herramientas utilizadas:

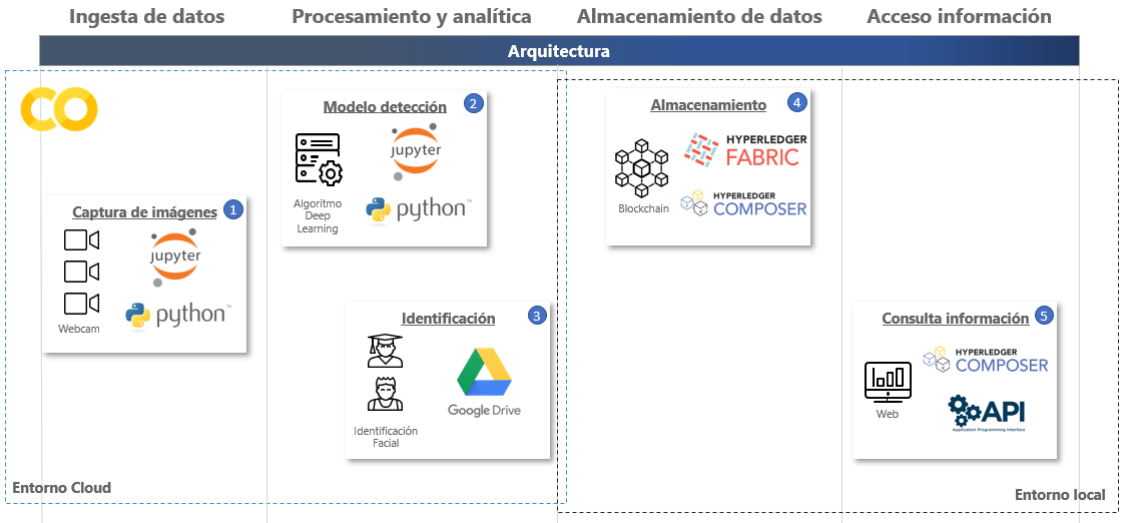


Figura 3.3: Arquitectura tecnológica del sistema

Fuente: Elaboración propia

1. A través de un notebook de Python ejecutado en Google Colaboratory se realiza la captura de las imágenes por una cámara web.
2. De manera análoga al punto anterior, mediante el procesamiento con GPUs proporcionado por Google Colaboratory se realiza el entrenamiento del modelo y se realiza la identificación facial.
3. Tras la identificación de las personas, se genera un fichero JSON que se almacena en Google Drive para solventar una limitación técnica del sistema que se detalla en la siguiente sección.
4. Desde el entorno local, se almacena la identificación detectada previamente en la cadena de bloques, bajo Hyperledger Fabric.
5. Por último, la información se puede consultar a través del consumo de un servidor API-REST implementado o, a través del entorno web Playground de Hyperledger Composer.

El detalle de los componentes desarrollados se aborda en las siguientes secciones de este mismo capítulo.

3.1.2. Flujograma del proceso

Con objeto de que el lector pueda comprender el flujo del sistema propuesto, en esta sección se presentan una serie de flujogramas que representan el funcionamiento del sistema de la manera más completa posible.

En primer lugar, se muestra un flujograma global y completo del sistema en el que se puede observar que, primero, se realiza la captura de la imagen para, posteriormente, aplicar el modelo de detección facial sobre ella. Tras identificar a la persona de manera correcta, se almacena en Hyperledger un registro con la información de dicha identificación:

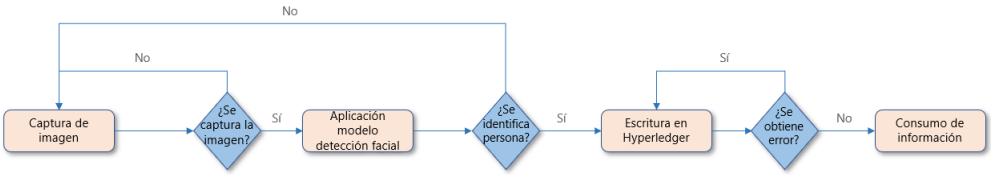


Figura 3.4: Flujograma esquemático global

Fuente: Elaboración propia

Sin embargo, para poder ejecutar el sistema desarrollado, antes es necesario realizar el despliegue de la solución y prepararla para su correcto funcionamiento. Los

siguientes dos flujogramas muestran este proceso, diferenciando la preparación del entorno Cloud y del entorno local.

En el entorno Cloud, en el que recordemos, se ejecuta la captura de las imágenes y la aplicación del modelo de detección facial, se debe preparar el modelo para que sea capaz de identificar a las personas deseadas. Si bien se utiliza un modelo pre-entrenado, como se detallará en la sección 3.2., primero es necesario realizar un pequeño entrenamiento del mismo con un conjunto de imágenes de las personas que se quieran identificar en el sistema.

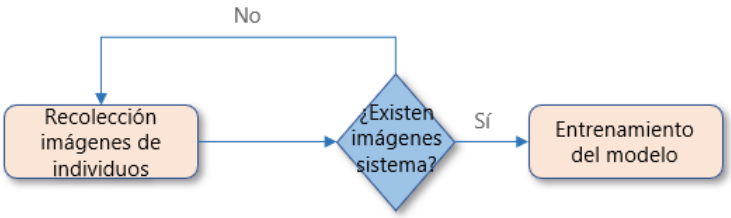


Figura 3.5: Flujograma preparación entorno Cloud

Fuente: Elaboración propia

Respecto al entorno local, en el que reside la infraestructura Blockchain para el almacenamiento de la información, es necesario realizar la definición y el despliegue de la arquitectura antes de ejecutar el sistema. El diseño y la definición de la red se analizará en detalle en la sección 3.3.

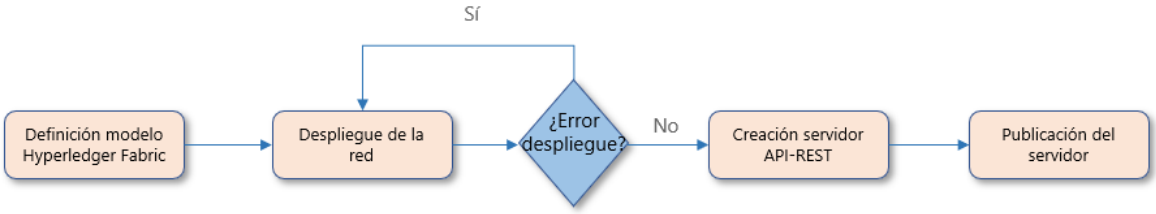


Figura 3.6: Flujograma preparación entorno local

Fuente: Elaboración propia

Cerrando esta sección, y para entender el funcionamiento estándar del sistema, se presenta un último flujograma. Éste representa una ejecución detallada del sistema desarrollado:

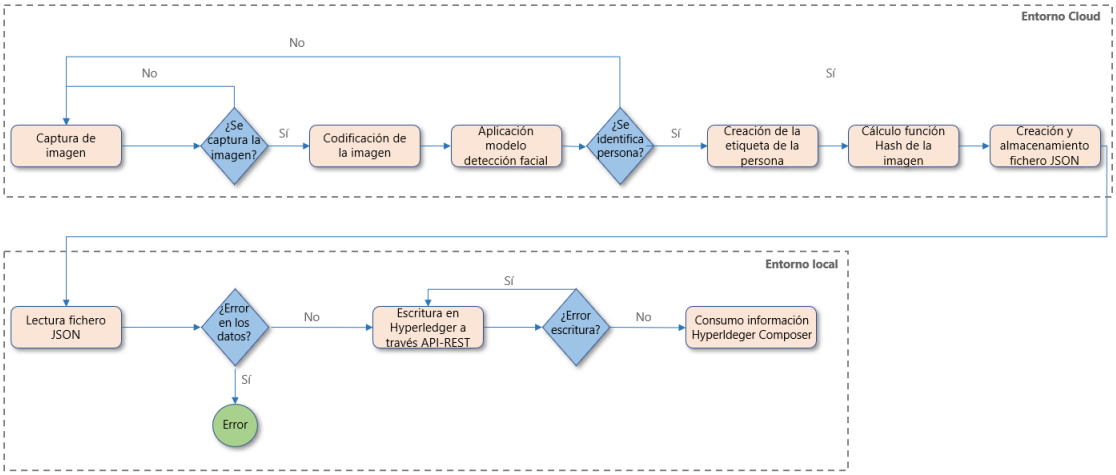


Figura 3.7: Flujograma detallado de ejecución estándar del sistema

Fuente: Elaboración propia

Como se puede observar, y para una mejor comprensión, el flujo de ejecución se explica a continuación:

1. Se **captura la imagen** a través de la cámara web.
2. Se **codifica** la imagen para que el modelo de detección facial la pueda procesar.
3. Se aplica el **modelo de detección facial** sobre la imagen codificada.
4. En caso de que se haya detectado de forma correcta al individuo, se **crea la etiqueta** de la persona y se aplica una **función hash** a la imagen para su posterior almacenamiento.
5. Se **crea un fichero en formato JSON** con la información que se almacenará en la cadena de bloques en el entorno Cloud para solventar una limitación técnica que se analiza en la siguiente sección.
6. Se **lee dicho fichero** en el entorno local y se obtiene la información que se desea almacenar.
7. Se **escribe el registro de la identificación en la cadena de bloques** implementada en Hyperledger Fabric a través de una API-REST creada para ello.
8. Por último, una vez se encuentra la información almacenada, ésta ya se puede **consumir** a través del entorno Playground de Hyperledger Composer o a través de la API-REST.

3.1.3. Limitaciones técnicas

Durante el desarrollo del sistema, se han encontrado una serie de limitaciones técnicas, en relación con el diseño planteado inicialmente, que han tenido que ser solventadas para poder realizar una prueba funcional del sistema. Esta sección enumera dichas limitaciones y detalla la solución adoptada para disminuir su impacto en el sistema. Las limitaciones encontradas han sido:

- A la hora de elegir el entorno donde se utiliza el modelo de detección facial se optó por un entorno Cloud, debido a que se necesita gran capacidad de cómputo y de memoria y no se disponía de máquinas locales donde poder implantar el modelo. Hay que tener en cuenta que el modelo de detección facial utiliza procesamiento GPU, ya que es mucho más eficiente para la resolución de este tipo de problemas. Por ello, para solventar la falta de recursos hardware, se adoptó una solución utilizando infraestructura en la nube, siendo la plataforma elegida Google Colaboratory.

- La elección de la plataforma Google Colaboratory provoca, sin embargo, otra limitación técnica. Desde el entorno Cloud no se puede acceder a la cámara web del ordenador que actúa como anfitrión, por lo que la captura de imágenes en tiempo real no puede ser realizada. Una alternativa era instalar una cámara web IP, pero no se disponía de una cámara web externa para ello. Por tanto, la solución que se adoptó fue el desarrollo de una función en JavaScript que permite la captura de imágenes (no vídeos) en el entorno Cloud a través de la cámara web del ordenador anfitrión.

- Un problema similar al anterior surge a la hora de escribir el registro de identificación en Hyperledger Fabric. La idea inicial era que, cada vez que se identificara una persona, el sistema automáticamente escribiera en Hyperledger Fabric. Sin embargo, el hecho de disponer de dos entornos diferenciados (Cloud y local) hace que esto no sea posible. Para poder realizar la escritura en Hyperledger Fabric, se ha desarrollado un servidor API-REST que permite realizar este tipo de tareas en el lenguaje de programación deseado a través de la API. Sin embargo, dicho servidor reside en el entorno local, por lo que no es accesible desde el entorno Cloud. Una alternativa habría sido implementar la arquitectura en el entorno Cloud, pero su uso no es gratuito. Por todo ello, para solventar este problema, en el momento en el que se identifica a una persona en el entorno Cloud se genera un fichero en formato JSON con la información que se desea registrar en la cadena de bloques y se almacena en Google Drive.

Desde el entorno local, se descarga dicho fichero de Google Drive y se registra la información en Hyperledger Fabric a través de la API creada para ello.

3.2. Tecnologías de desarrollo

La implementación de los componentes del sistema desarrollado se ha llevado a cabo con diversas tecnologías cuya elección se debe a los requerimientos de los componentes del propio sistema. A continuación, se define cada una de ellas, indicando los componentes donde se han utilizado:

- **Python**. Lenguaje interpretado de scripting de código abierto orientado a objetos. Creado por Guido van Rossum, está preparado para el desarrollo de manera independiente a la plataforma y al sistema operativo. Permite desarrollar, desde aplicaciones de escritorio hasta aplicaciones web. Se trata de un tipo de lenguaje más flexible que los lenguajes compilados. Internamente, emplea un intérprete donde se traduce el código fuente a bytecode la primera vez que se ejecuta. En los últimos años, se ha hecho muy popular gracias a todas las ventajas que proporciona. Durante el desarrollo del sistema, Python se ha utilizado para implementar el modelo de detección facial.

- **GNU Bash**. Es un intérprete y procesador de comandos shell que permite ejecutar scripts por consola para automatizar tareas. Fue desarrollado en el año 1987 para el proyecto GNU. Actualmente, es el lenguaje estándar de todas las distribuciones GNU/Linux. En el sistema, se ha utilizado para la instalación de la infraestructura Blockchain de Hyperledger Fabric.

- **JavaScript**. Es un lenguaje de programación interpretado y orientado a objetos muy ligero. Está basado en prototipos, es imperativo y está débilmente tipado. Surge por la necesidad de incrementar las funcionalidades de los sistemas web del lado del cliente, incluyendo contenido dinámico y de mayor complejidad. Durante el desarrollo del sistema, se ha utilizado para desarrollar la red de negocio Blockchain de Hyperledger Composer, ya que es un lenguaje impuesto por esta tecnología para diseñar la lógica de la red. También se ha utilizado para poder capturar imágenes a través de la cámara web desde el entorno Cloud.

- **Hyperledger Composer Modeling Language (CTO)**. Es un lenguaje de programación orientado a objetos, incluido en la tecnología Hyperledger Composer, y utilizado para definir el modelo de la red Hyperledger. En el

sistema se ha usado precisamente para eso, para definir el modelo de la red Blockchain.

3.3. Modelo de reconocimiento facial

Como se ha comentado al principio de este capítulo, el sistema desarrollado está formado por dos grandes bloques: el modelo de reconocimiento facial y la implementación de Hyperledger. Esta sección analiza en detalle el primer bloque y expone el trabajo realizado.

Definición teórica

En primer lugar, es importante entender por qué Deep Learning funciona con tan buenos resultados en el problema del reconocimiento facial. El secreto está en una técnica denominada 'Deep Metric Learning'. En los problemas de Deep Learning tradicionales, por lo general, se entrena una red para aceptar una única imagen como entrada y proporcionar como salida una etiqueta para dicha imagen. A la hora de abordar el problema de detección facial, en lugar de obtener como resultado una etiqueta para la imagen, se va a obtener un vector de características con valores reales.

En este proyecto, el modelo se ha aplicado mediante Python y se han utilizado las librerías «dlib» y «facial_recognition». La librería «dlib» proporciona un modelo pre-entrenado con un rendimiento muy bueno que se detallará más adelante. También se sirve de un vector de 128 números reales, que son utilizados para «cuantificar» la cara.

El modelo utilizado en el desarrollo del sistema está basado en la red ResNet-34 diseñada por K. He, X. Zhang, S. Ren y J. Sun [58], pero con menos capas y reduciendo a la mitad el número de filtros. En concreto, el modelo es una red ResNet con 29 capas convolucionales. El diseño y el entrenamiento de la red ha sido realizado por Davis King sobre un conjunto de datos de, aproximadamente, tres millones de imágenes procedentes de varios datasets públicos. En el dataset creado por Davis King se encuentran 7485 individuos diferentes.

Probando el modelo sobre el conjunto de datos «Labeled Faces in the Wild (LFW)» [35] se obtiene una precisión de 99.38 %. Esto significa que, dadas dos caras diferentes, el modelo predice correctamente si las imágenes son de la misma persona el 99.38 % de las veces.

El desarrollo del modelo por parte de Davis King está basado en su librería «dlib». El modelo pre-entrenado es de dominio público y puede ser consultado en GitHub [59].

De manera resumida, se puede decir que el proceso de entrenamiento del modelo se realiza utilizando combinaciones de tres caras distintas, perteneciendo dos a la misma persona. La red genera un vector de 128 valores para cada una de las tres caras. Con las dos caras de la misma persona, se ajustan los pesos de la red para hacer que ambos vectores se encuentren más cerca entre ellos y más alejados de la otra cara (esta distancia se puede medir con varias métricas distintas). Para más detalle se puede consultar la librería de Davis King [59].

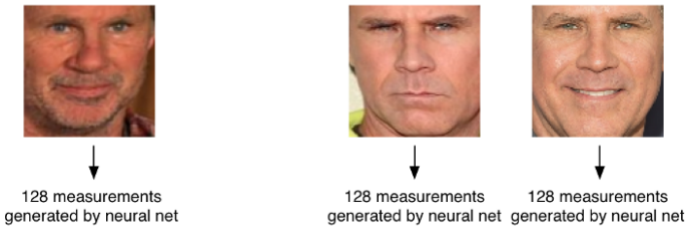


Figura 3.8: Proceso entrenamiento del modelo

Fuente:

`https://www.pyimagesearch.com/2018/06/18/face-recognition-with-opencv-python-and-deep-learning/`

Antes de continuar con la implementación práctica del sistema, es importante indicar que existen dos pasos fundamentales a la hora de resolver el problema de la detección facial:

- En primer lugar, aunque parezca obvio, se debe detectar las caras de una imagen antes de tratar de identificarlas. Para tratar este problema, normalmente, se convierte la imagen a blanco y negro y se recorren todos los píxeles de la imagen, identificando en cada uno de ellos los píxeles que la rodean directamente. El objetivo principal es determinar cuan oscuros son los píxeles que rodean a otro e identificar mediante una flecha la dirección y el sentido de la luz. Realizando esta tarea con todos los píxeles de manera independiente, se obtiene un patrón que se compara con patrones de caras previamente definidos.

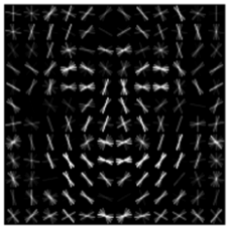


Figura 3.9: Patrón de detección facial

Fuente:

https://www.pyimagesearch.com/2018/06/18/face-recognition-with-opencv-python-and-deep-learning/

- El segundo paso, como se ha comentado al principio de la sección, es codificar las imágenes. Para ello, se necesita extraer varias medidas básicas de cada cara. De esta manera, se podrán comparar las mismas medidas en todos los casos y determinar, así, si dos caras pertenecen a la misma persona o no. Ejemplos de estas medidas son: el tamaño de cada oreja, el espacio entre los ojos, la longitud de la nariz, etc.

  Para los humanos, las medidas básicas de las caras parecen obvias (el color de los ojos, por ejemplo), por lo que, en principio, no tendría sentido mirar pixel por pixel cada una de las imágenes. Sin embargo, varios estudios han concluido que es mucho mejor dejar al ordenador determinar las medidas básicas de una cara que hacer este trabajo de manera manual. Este es uno de los puntos donde Deep Learning ofrece una de sus ventajas más importantes, ya que realiza un mejor trabajo descubriendo qué partes de la cara son las más importantes que el que pueda determinar un humano.

  Esta codificación requiere de gran capacidad de cómputo, aunque solamente es necesario realizarlo una vez. Dicha codificación forma un vector de 128 valores, con las medidas más importantes de una cara. Sin embargo, la principal desventaja de este método es que se desconoce qué medidas se están teniendo en cuenta.

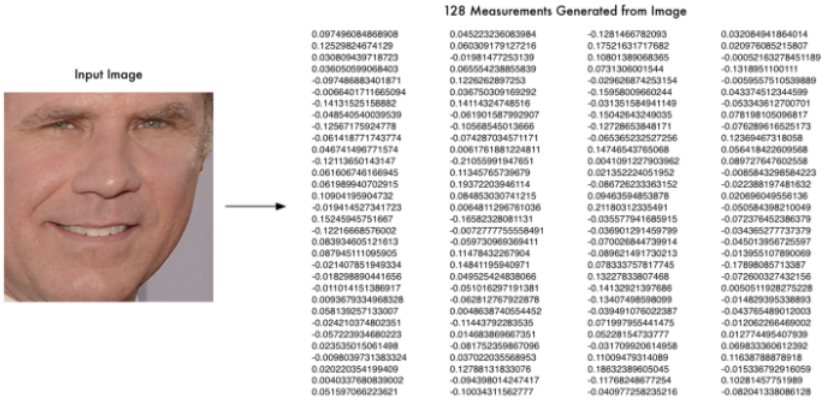


Figura 3.10: Codificación de una imagen

Fuente:

`https://www.pyimagesearch.com/2018/06/18/face-recognition-with-opencv-python-and-deep-learning/`

Implementación práctica

Como se ha comentado en esta sección, la implementación práctica de este bloque se ha realizado en el entorno Cloud Google Colaboratory. Dentro de esta plataforma, se ha creado un notebook de Python donde se incluye todo el desarrollo realizado.

En este apartado se detalla la implementación realizada y se muestran extractos del código fuente del sistema. El proyecto se encuentra publicado en GitHub y se puede consultar [60].

En primer lugar, se realiza la instalación y la importación de las librerías requeridas:

```
from imutils import paths
import face_recognition
import argparse
import pickle
import cv2
import os
```

Listing 3.1: Importación librerías

A continuación, es necesario conectarse con Google Drive, ya que es ahí donde reside el conjunto de datos de entrenamiento del modelo y algunas imágenes de pruebas. El dataset está conformado por 160 imágenes correspondientes a 16 individuos diferentes. Entre estos individuos, quince son personajes famosos y uno no lo es. En Google Drive, también se almacena el fichero con todas estas imágenes codificadas (como se verá más adelante) y el fichero en formato JSON generado para poder almacenar la información en Hyperledger.

```
from google.colab import drive
drive.mount('/content/gdrive')
```

Listing 3.2: Importación Google Drive

El siguiente paso es codificar las imágenes y transformarlas en vectores de 128 valores, previa identificación de las caras en las imágenes. El fichero con este resultado se almacena en Google Drive.

```
DATASET = '/content/gdrive/My Drive/dataset'
rutaImagenes = list(paths.list_images(DATASET))

encodingsConocidos = []
nombresConocidos = []

for (i, imagePath) in enumerate(rutaImagenes):
  nombre = imagePath.split(os.path.sep)[-2]

  imagen = cv2.imread(imagePath)
  rgb = cv2.cvtColor(imagen, cv2.COLOR_BGR2RGB)

  caja = face_recognition.face_locations(rgb,model='cnn')

  encodings = face_recognition.face_encodings(rgb, caja)

  for encoding in encodings:
    encodingsConocidos.append(encoding)
    nombresConocidos.append(nombre)

data = {"encodings": encodingsConocidos, "names": nombresConocidos}
f = open("/content/gdrive/My Drive/encodings.pickle", "wb")
f.write(pickle.dumps(data))
f.close()
```

Listing 3.3: Codificación de las imágenes

En este momento, el modelo ya se encuentra preparado para poder realizar identificaciones faciales de los individuos registrados en el sistema. A la hora de aplicar el modelo, se han implementado tres aproximaciones:

- Detección facial en imágenes.
- Detección facial en vídeos.
- Detección facial en imágenes capturadas a través de la cámara web.

Dado que, únicamente, la tercera opción es la que aplica al sistema de identificación desarrollado, ésta será la que se muestre en detalle, presentando su implementación.

A la hora de poder capturar imágenes desde la cámara web en el entorno de Google Colaboratory, es necesario desarrollar una función en JavaScript y llamarla desde Python para solucionar esta limitación técnica:

```
from IPython.display import display, Javascript
from google.colab.output import eval_js
from base64 import b64decode
from IPython.display import Image

def take_photo(filename='/content/gdrive/My Drive/photo.jpg',
    quality=0.8):
  js = Javascript('''
    async function takePhoto(quality) {
      const div = document.createElement('div');
      const capture = document.createElement('button');
      capture.textContent = 'Capture';
      div.appendChild(capture);

      const video = document.createElement('video');
      video.style.display = 'block';
      const stream = await navigator.mediaDevices.getUserMedia({
    video: true});

      document.body.appendChild(div);
      div.appendChild(video);
      video.srcObject = stream;
      await video.play();

      google.colab.output.setIframeHeight(document.documentElement.
    scrollHeight, true);

      await new Promise((resolve) => capture.onclick = resolve);

      const canvas = document.createElement('canvas');
      canvas.width = video.videoWidth;
      canvas.height = video.videoHeight;
      canvas.getContext('2d').drawImage(video, 0, 0);
      stream.getVideoTracks()[0].stop();
      div.remove();
      return canvas.toDataURL('image/jpeg', quality);
    }
```

```
    ''')
  display(js)
  data = eval_js('takePhoto({})'.format(quality))
  binary = b64decode(data.split(',')[1])
  with open(filename, 'wb') as f:
    f.write(binary)
  return filename


print('Inicializando la camara web...')

try:
  vs = take_photo()
  writer = None
  time.sleep(2.0)
  display(Image('/content/gdrive/My Drive/photo.jpg'))
except Exception as err:
  print(str(err))
```

Listing 3.4: Captura cámara web

Como se puede ver en el código, se habilita un botón en el Notebook de Python a través del cual se puede capturar una foto. Esta foto se guarda en Google Drive.

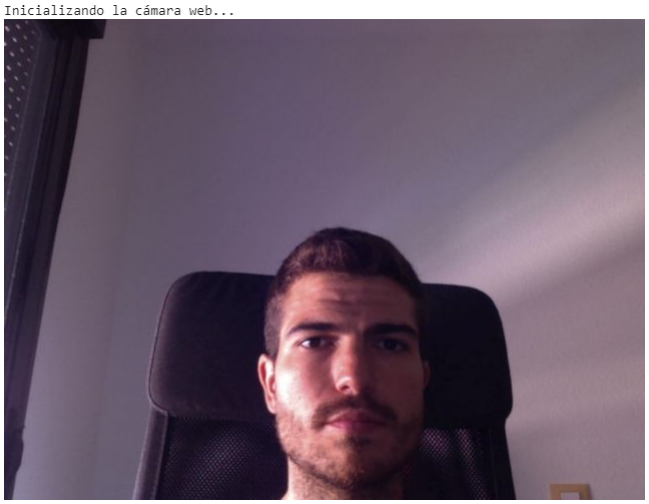


Figura 3.11: Imagen captura a través de la cámara web

Fuente: Elaboración propia

A continuación, es necesario codificar la imagen de manera análoga al proceso anterior.

```
print("Carga de codificaciones...")
```

```
data = pickle.loads(open("/content/gdrive/My Drive/encodings.pickle
    ", "rb").read())

imagen = cv2.imread("/content/gdrive/My Drive/photo.jpg")
rgb = cv2.cvtColor(imagen, cv2.COLOR_BGR2RGB)

print("Reconociendo caras...")
cajas = face_recognition.face_locations(rgb, model='cnn')
encodings = face_recognition.face_encodings(rgb, cajas)
nombres = []
```

Listing 3.5: Codificación de la imagen

En este momento es cuando el modelo cumple su función y resuelve el problema de la identificación facial. Para ello, comprueba el vector de la imagen codificada con el resto de los vectores creados previamente. Dicha comparación se basa en el cálculo de la distancia Euclidea entre ambos vectores. Si dicha distancia se encuentra por debajo de un umbral, se considera que ambas caras son iguales.

```
for encoding in encodings:
  coincidencias = face_recognition.compare_faces(data["encodings"],
    encoding)
  nombre = "Desconocido"

  if True in coincidencias:
    matchedIdxs = [i for (i, b) in enumerate(coincidencias) if b]
    counts = {}

    for i in matchedIdxs:
      nombre = data["names"][i]
      counts[nombre] = counts.get(nombre, 0) + 1

    nombre = max(counts, key=counts.get)

  nombres.append(nombre)
```

Listing 3.6: Identificación facial

Como se puede ver, el modelo identifica de manera correcta al individuo:

Figura 3.12: Identificación facial detectada

Fuente: Elaboración propia

Por último, quedaría pendiente calcular la función hash de la imagen y generar el fichero JSON con la información que se va a almacenar en Hyperledger. Esta información se compone de: identificación de la persona, valor Hash de la imagen y timestamp. La función hash que se utiliza es «average hash» [61] de la librería de Python «imagehash» [62].

```
from PIL import Image
import imagehash
import datetime
import json

hash_img = imagehash.average_hash(Image.open('/content/gdrive/My
    Drive/examples/borja2.jpg'))
etiqueta = nombres[0]
task = {"$class": "org.basic.identificacion.FacialRecognition","
    registro": "org.basic.identificacion.Registro#REG_001", "fecha":
     str(datetime.datetime.now()),"etiqueta": etiqueta,"hash_imagen"
    :str(hash_img)}

f = open("/content/gdrive/My Drive/identificacion.json", "w")
f.write(json.dumps(task))
f.close()
```

Listing 3.7: Generación y almacenamiento fichero JSON


```
{ "$class": "org.basic.identificacion.FacialRecognition",
  "registro": "org.basic.identificacion.Registro#REG_001",
  "fecha": "2019-08-10 11:38:19.571525",
  "etiqueta": "Borja",
  "hash_imagen": "ffffff900000000"
}
```


Figura 3.13: Ejemplo fichero JSON

Fuente: Elaboración propia

3.4. Implementación de la cadena de bloques

En esta sección se analiza y detalla la implementación del segundo gran bloque del proyecto: el desarrollo de la infraestructura basada en cadena de bloques para almacenar la información de las identificaciones realizadas.

Definición teórica

Si bien, actualmente, existen multitud de tecnologías Blockchain diferentes, el problema que trata de resolver el sistema desarrollado busca una solución que sea privada, ya que almacenar la información de diferentes individuos en una red pública puede suponer problemas legales relativos a la protección de datos. Además, el carácter del sistema es para un ámbito empresarial, donde únicamente sea una entidad y sus participantes quienes puedan acceder a los datos. Por ello, la solución Blockchain adoptada es Hyperledger.

Hyperledger proporciona un enfoque modular donde se ubican todos los componentes que lo conforman. En un primer nivel, se encuentra la infraestructura que hace referencia al ecosistema y, en un segundo nivel, es donde están los frameworks que proporcionan los elementos necesarios para el despliegue de la cadena de bloques (como el algoritmo de consenso, los contratos inteligentes, etc.).

Antes de continuar con la implementación práctica de la solución, es muy importante entender el flujo de Hyperledger, así como sus componentes principales. En primer lugar, en Hyperledger existen tres tipos de nodos diferentes:

- Endorser peer: reciben las peticiones de transacciones desde el cliente y aprueba o deniegan la transacción.
- Anchor peer: habilitan la comunicación entre diferentes nodos de la misma organización.
- Ordering peer: se encargan de crear los bloques, enviarlos al resto de nodos y hacer que la información persista, actualizando el ledger.

En el momento que se ejecuta una transacción en Hyperledger Fabric, el flujo es el siguiente [63]:

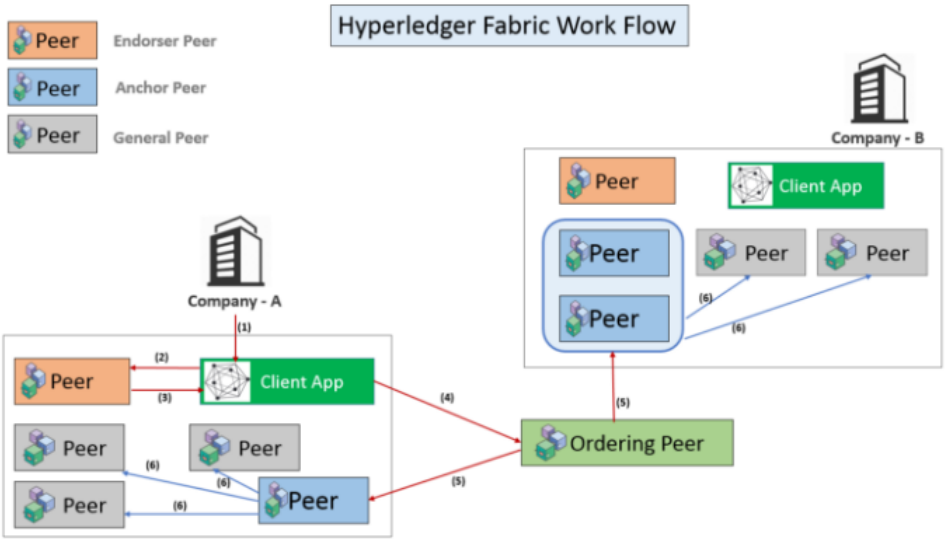


Figura 3.14: Flujo de proceso de Hyperledger Fabric

Fuente: Hyperledger

Donde cada número representa:

1. Un miembro de la organización solicita realizar una transacción a través de una aplicación cliente.
2. El cliente emite la solicitud de transacción a varios endorser peers.
3. Cada uno de estos nodos realiza varias comprobaciones y hace una simulación del Smart Contract implementado en la red, devolviendo los resultados firmados al cliente con la aprobación o denegación de la transacción.
4. El cliente envía la transacción aprobada al ordering peer para que se incluya un nuevo bloque en el ledger. Este bloque se distribuye a todos los anchor peers.
5. Los anchor peers envían este nuevo bloque al resto de nodos de la red para que actualicen su ledger y la red quede actualizada y sincronizada.

Por otro lado, para realizar el despliegue de la arquitectura en Hyperledger Fabric, se ha utilizado Hyperledger Composer, que dispone de un conjunto de abstracciones que permiten modelar e integrar una solución Blockchain de manera sencilla, haciendo que el desarrollador dedique menos tiempo en el propio despliegue y se centre en la definición del modelo y en la lógica de la red:

- La definición del modelo de negocio hace referencia al modelo de datos de la red. Está formado por: activos, participantes, transacciones y eventos; definiendo, a su vez, la relación entre estos elementos.
- La lógica de la red se define en un fichero JavaScript dentro de Hyperledger Composer. Este componente se considera como los chaincode de la red y se despliega y ejecuta en cada nodo de la cadena de bloques.

Po otro lado, respecto al modelo de negocio de Hyperledger, es importante comprender los conceptos claves que forman una red, ya que se hablará de ellos en la implementación práctica. Estos elementos son:

- Activos. Se refiere a bienes, servicios o propiedades almacenados en registros de la cadena de bloques. Representa cualquier elemento que pueda ser utilizado en la red de negocio.
- Participantes. Hace referencia a los miembros de la red de negocio. Los participantes poseen activos y, a través de las transacciones, los intercambian y modifican sus propiedades.
- Transacciones. Representan las reglas de negocio de la red. Son las reglas a las que se acogen todos los participantes.

Además, Hyperledger Composer ofrece una herramienta web (Composer Playground) donde poder probar el despliegue de una Blockchain o donde consultar los datos de la misma. También ofrece la creación de un servidor API-REST para que la red pueda ser consumida desde soluciones externas. Ambos elementos se utilizan en el desarrollo del sistema.

Implementación práctica

A la hora de realizar la implementación práctica, el primer paso es desplegar la arquitectura de Hyperledger en el entorno local. Para ello, la documentación propia de la tecnología lista una serie de prerrequisitos que deben encontrarse en el sistema. En concreto, es necesario que se encuentre instalado: Node.js, Docker y Python.

Para cumplir estos requisitos, se ofrece un script que instala todas las aplicaciones necesarias:

```
curl -O -k https://hyperledger.github.io/composer/latest/prereqs-
    ubuntu.sh
chmod u+x prereqs-ubuntu.sh
```

Listing 3.8: Instalación prerequisitos Hyperledger

Adicionalmente, se debe instalar Hyperledger Composer, junto a su consola de comandos, al servidor REST que se implementará y al componente Playground:

```
npm install -g composer-cli@0.20
npm install -g composer-rest-server@0.20
npm install -g generator-hyperledger-composer@0.20
npm install -g yo
npm install -g composer-playground@0.20
```

Listing 3.9: Instalación Hyperledger Composer

Por último, quedaría instalar Hyperledger Fabric, que es donde residirá la cadena de bloques:

```
mkdir ~/fabric-dev-servers && cd ~/fabric-dev-servers

curl -O https://raw.githubusercontent.com/hyperledger/composer-
    tools/master/packages/fabric-dev-servers/fabric-dev-servers.tar.
    gz
tar -xvf fabric-dev-servers.tar.gz

export FABRIC_VERSION=hlfv12
./downloadFabric.sh

cd ~/fabric-dev-servers
 export FABRIC_VERSION=hlfv12
 ./startFabric.sh
 ./createPeerAdminCard.sh
```

Listing 3.10: Instalación Hyperledger Fabric

```
borja@borja-VirtualBox:~/fabric-dev-servers$ ./createPeerAdminCard.sh
Development only script for Hyperledger Fabric control
Running 'createPeerAdminCard.sh'
FABRIC_VERSION is set to 'hlfv12'
FABRIC_START_TIMEOUT is unset, assuming 15 (seconds)

Using composer-cli at v0.20.8

Successfully created business network card file to
        Output file: /tmp/PeerAdmin@hlfv1.card

Command succeeded


Successfully imported business network card
        Card file: /tmp/PeerAdmin@hlfv1.card
        Card name: PeerAdmin@hlfv1

Command succeeded

The following Business Network Cards are available:

Connection Profile: hlfv1
| Card Name        | UserId      | Business Network |
| PeerAdmin@hlfv1  | PeerAdmin   |                  |
```

Figura 3.15: Instalación Hyperledger Fabric

Fuente: Elaboración propia

En este momento ya estaría el entorno local preparado, por lo que se puede comenzar a definir la red de negocio de Hyperledger Composer.

La definición de la red de negocio es la parte más importante en cualquier cadena de bloques, ya que representa la conversión al entorno digital de cualquier de los elementos que conforma la red y es donde se encuentra toda la lógica de negocio. Para definir la red de negocio del sistema se ha utilizado Hyperledger Composer. La red se despliega a través de un fichero .bna (Business Network Archive) y es mantenida en todos los nodos de Hyperledger Fabric. Si bien Hyperledger Composer no permite explotar todas las posibilidades que existen en Hyperledger Fabric, cumple con los requisitos del sistema desarrollado y proporciona grandes ventajas, siendo fundamental la simplificación del proceso de implementación de la cadena de bloques.

Antes de continuar con la definición de la red de negocio construida, es importante entender el acoplamiento de Hyperledger Composer en Hyperledger Fabric. El siguiente diagrama, obtenido de la documentación propia de Hyperledger [52], es muy explicativo.

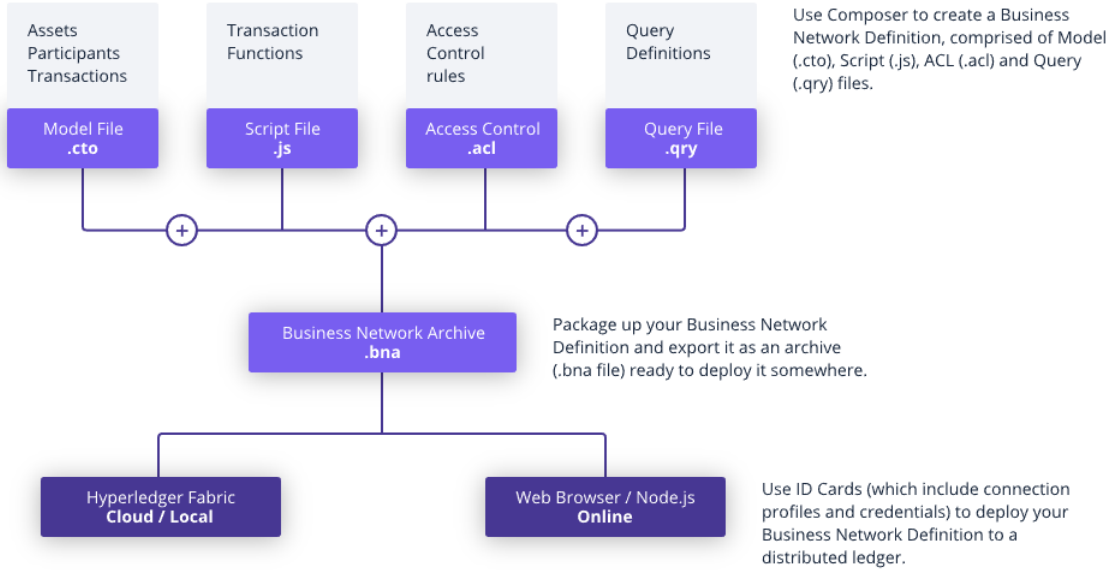


Figura 3.16: Arquitectura Hyperledger

Fuente: Hyperledger

Volviendo al dominio del sistema desarrollado, se ha determinado el sistema de nombres org.basic.identificacion y se han definido los siguientes elementos:

- Participantes:
  - Empleado. Representa a un individuo que es un empleado de una empresa. Tiene los siguientes atributos:
    - dni: identificador único del empleado.
    - email: correo electrónico del empleado.
    - balanceHoras: contabiliza el número total de horas realizadas por un empleado.
  - Empresa. Representa a una empresa. Tiene los siguientes atributos:
    - cif: identificador único de la empresa.
    - email: correo electrónico de la empresa.
- Activos:
  - Registro. Representa cada una de las identificaciones faciales realizadas. Tiene los siguientes atributos:
    - registroId: identificador del registro.

    - estado: tipo abstracto de datos definido que puede tomar los valores: 'OK', 'ERR' o 'PTE', permitiendo al usuario confirmar o rechazar su identificación.
    - count: contabiliza el número de identificaciones realizadas para cada participante.
    - facialrecognitions: vector con todas las transacciones de tipo FacialRecognition realizadas.
    - contract: contrato al que hace referencia el registro.
  - Contrato. Representa el contrato entre participantes de la red. Tiene los siguientes atributos:
    - contractId: identificador del contrato.
    - empleado: representa uno de los participantes del contrato de tipo Empleado.
    - empresa: representa uno de los participantes del contrato de tipo Empresa.
    - fecha: fecha en la que se establece el contrato.
    - hora_entrada: hora de entrada del empleado en la empresa.
    - hora_salida: hora de salida del empleado en la empresa.
- Transacciones:
  - FacialRecognition. Transacción básica de la red. Es la transacción que registra cualquier identificación facial realizada. Tiene los siguientes atributos:
    - registro: enlace con un Registro donde se almacenan las identificaciones.
    - fecha: fecha y hora de la identificación realizada.
    - etiqueta: nombre del individuo identificado.
    - hash_imagen: Hash de la imagen utilizada para realizar la identificación.
  - ConfirmarIdentificacion. Confirma una identificación realizada por el sistema. Sólo tiene un único atributo de tipo Registro.

- RechazarIdentificacion. Rechaza una identificación realizada por el sistema. Sólo tiene un único atributo de tipo Registro.
- setupProject. Ejecuta el sistema desarrollado e inicializa todos los elementos de la cadena de bloques.

La topología descrita y todos los componentes se definen en un fichero CTO.

```
1
2 namespace org.basic.identificacion
3
4 enum Status {
5   o OK
6   o ERR
7   o PDTE
8 }
9
10 participant Empleado identified by dni {
11   o String dni
12   o String email
13   o Double balanceHoras
14 }
15
16 participant Empresa identified by cif {
17   o String cif
18   o String email
19 }
20
21
22 transaction FacialRecognition {
23   --> Registro registro
24   o DateTime fecha
25   o String etiqueta
26   o String hash_imagen
27 }
28
29 transaction ConfirmarIdentificacion{
30   --> Registro registro
31 }
32
33 transaction RechazarIdentificacion{
34   --> Registro registro
35 }
36
37 asset Registro identified by registroId {
38   o String registroId
```

```
39   o Status estado
40   o Double count
41   o FacialRecognition[] facialrecognitions optional
42   --> Contract contract
43 }
44
45
46 asset Contract identified by contractId {
47   o String contractId
48   --> Empleado empleado
49   --> Empresa empresa
50   o DateTime fecha
51   o Double hora_entrada
52   o Double hora_salida
53 }
54
55
56 transaction setupProject {
57 }
```

Listing 3.11: Definición de los elementos de la red en Hyperledger Composer

Las transacciones indicadas representan las reglas de negocio de la red y se definen dentro del modelo bajo un fichero desarrollado en JavaScript denominado 'logic.js'. Las funciones implementadas son las siguientes:

- facialRecognition: encargada de registrar las identificaciones faciales realizadas. Actualiza el contador del número de identificaciones realizadas y establece el estado del registro como 'PTE'.
- confirmarIdentificacion: utilizada para confirmar la identificación realizada de un registro. Actualiza el estado del registro a 'OK'.
- rechazarIdentificacion: deniega la identificación del registro y actualiza su estado a 'ERR'.
- setupProject: inicializa el proyecto. Para ello, crea dos participantes (un empleado y una empresa) con sus atributos básicos, un contrato entre ellos y un registro con el vector de identificaciones faciales vacío.

```
1
2 /**
3  * Facial recognition made by the model
4  * @param {org.basic.identificacion.FacialRecognition}
    FacialRecognition - the FacialRecognition transaction
```

```
 * @transaction
 */
async function facialRecognition(facialRecognition) {  // eslint-disable-line no-unused-vars

    const NS = 'org.basic.identificacion';
    const registro = facialRecognition.registro;
    const contract = registro.contract;
    const factory = getFactory();

    console.log('Identificacion facial realizada ' + facialRecognition.etiqueta );

    if (registro.facialrecognitions) {
        registro.facialrecognitions.push(facialRecognition);
    } else {
        registro.facialrecognitions = [facialRecognition];
    }

    registro.estado = 'PDTE';
    registro.count = registro.facialrecognitions.length;

    const registroRegistry = await getAssetRegistry(NS + '.Registro');
    await registroRegistry.update(registro);
}

/**
 * Facial recognition confirmation
 * @param {org.basic.identificacion.ConfirmarIdentificacion} ConfirmarIdentificacion - the FacialRecognition confirmation transaction
 * @transaction
 */
async function confirmarIdentificacion(confirmarIdentificacion) {  // eslint-disable-line no-unused-vars

    const NS = 'org.basic.identificacion';
    const registro = confirmarIdentificacion.registro;
    const factory = getFactory();

    console.log('Identificacion facial confirmada ');

    registro.estado = 'OK';

```

```
44     const registroRegistry = await getAssetRegistry(NS + '.Registro
    ');
45     await registroRegistry.update(registro);
46 }
47
48 /**
49  * Facial recognition confirmation
50  * @param {org.basic.identificacion.RechazarIdentificacion}
    RechazarIdentificacion - the FacialRecognition deny transaction
51  * @transaction
52  */
53 async function rechazarIdentificacion(rechazarIdentificacion) {  //
    eslint-disable-line no-unused-vars
54
55     const NS = 'org.basic.identificacion';
56     const registro = rechazarIdentificacion.registro;
57     const factory = getFactory();
58
59     console.log('Identificacion facial rechazada ');
60
61     registro.estado = 'ERR';
62
63     var idEvent = factory.newEvent(NS, 'EventoRechazarId');
64     idEvent.registro = registro;
65     idEvent.estado = 'ERR';
66     idEvent.mensaje = 'Identificacion rechazada';
67     console.log(idEvent.message);
68     emit(idEvent);
69
70     const registroRegistry = await getAssetRegistry(NS + '.Registro
    ');
71     await registroRegistry.update(registro);
72 }
73
74 /**
75  * Initialize some test assets and participants useful for running
    the projects.
76  * @param {org.basic.identificacion.setupProject} SetupProject -
    the SetupProject transaction
77  * @transaction
78  */
79
80 async function setupProject(setupProject) {
81
82     const factory = getFactory();
```

```
83     const NS = 'org.basic.identificacion';
84
85     const empleado = factory.newResource(NS, 'Empleado', '12045484F
    ');
86     empleado.email = 'borja@email.com';
87     empleado.balanceHoras = 0;
88
89     const empresa = factory.newResource(NS, 'Empresa', 'A80569352')
    ;
90     empresa.email = 'borja@email.com';
91
92     const contract = factory.newResource(NS, 'Contract', 'CON_002')
    ;
93     contract.empleado = factory.newRelationship(NS, 'Empleado', '
    12045484F');
94     contract.empresa = factory.newRelationship(NS, 'Empresa', '
    A80569352');
95     const fecha = setupProject.timestamp;
96     fecha.setDate(fecha.getDate());
97     contract.fecha = fecha;
98     var d = new Date(2019, 08, 10, 08, 00, 00, 0);
99     var n = d.getHours();
100    contract.hora_entrada = n;
101    var d = new Date(2019, 08, 10, 15, 00, 00, 0);
102    var n = d.getHours();
103    contract.hora_salida = n;
104
105    const registro = factory.newResource(NS, 'Registro', 'REG_001')
    ;
106    registro.estado = 'OK';
107    registro.count = 1;
108    registro.contract = factory.newRelationship(NS, 'Contract', '
    CON_002');
109
110    const empleadosRegistry = await getParticipantRegistry(NS + '.
    Empleado');
111    await empleadosRegistry.addAll([empleado]);
112
113    const empresaRegistry = await getParticipantRegistry(NS + '.
    Empresa');
114    await empresaRegistry.addAll([empresa]);
115
116    const contractRegistry = await getAssetRegistry(NS + '.Contract
    ');
117    await contractRegistry.addAll([contract]);
```

```
118
119     const registroRegistry = await getAssetRegistry(NS + '.Registro
    ');
120     await registroRegistry.addAll([registro]);
121 }
```

Listing 3.12: Definición de las transacciones

Una vez definidos los componentes de la red y las reglas de negocio de las mismas, se puede realizar su despliegue. Para ello, se debe generar un fichero .bna.

```
1 composer archive create -t dir -n .
```

Listing 3.13: Creación del fichero de red

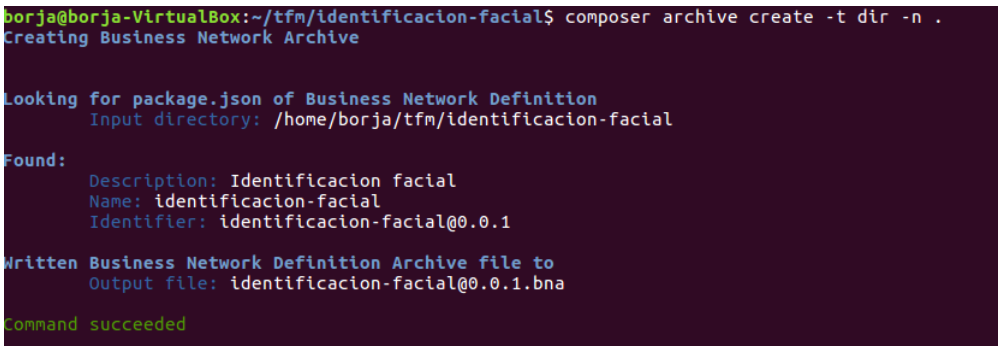


Figura 3.17: Generación de la red Hyperledger

Fuente: Elaboración propia

El último paso es realizar el despliegue y la instalación de la red en Hyperledger Fabric:

```
1 composer network install --card PeerAdmin@hlfv1 --archiveFile
    identificacion-facial@0.0.1.bna
2
3 composer network start --networkName identificacion-facial --
    networkVersion 0.0.1 --networkAdmin admin --
    networkAdminEnrollSecret adminpw --card PeerAdmin@hlfv1 --file
    networkadmin.card
4
5 composer card import --file networkadmin.card
```

Listing 3.14: Despliegue e instalación de la red

```
borja@borja-VirtualBox:~/tfm/identificacion-facial$ composer network start --networkName identi
ficacion-facial --networkVersion 0.0.1 --networkAdmin admin --networkAdminEnrollSecret adminpw
--card PeerAdmin@hlfv1 --file networkadmin.card
Starting business network identificacion-facial at version 0.0.1

Processing these Network Admins:
        userName: admin

✔ Starting business network definition. This may take a minute...
Successfully created business network card:
        Filename: networkadmin.card

Command succeeded
```

Figura 3.18: Despliegue y lanzamiento de la red Hyperledger

Fuente: Elaboración propia

Se puede comprobar que la red se ha desplegado de manera correcta en Hyperledger Composer Playground:

```
composer-playground
```

Listing 3.15: Despliegue e instalación de la red

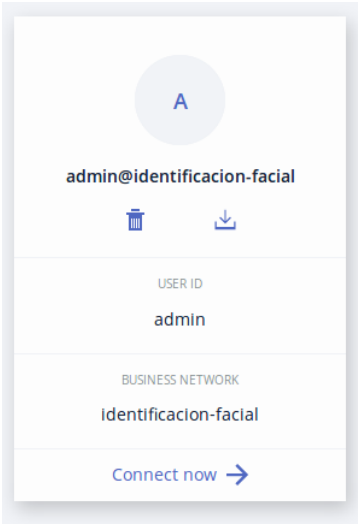


Figura 3.19: Comprobación de la red en Hyperledger Composer Playground

Fuente: Elaboración propia

3.5. Implementación de la solución global

Tras la implementación de los dos bloques principales del sistema, en esta sección se aborda cómo se ha realizado la integración de ambas partes y cómo se ha conformado el sistema global desarrollado.

En primer lugar, es necesario crear una API de Hyperledger Fabric que permita interaccionar con la cadena de bloques desde servicios externos. Para ello, se genera un servidor REST mediante Hyperledger Composer:

```
composer-rest-server
```

Listing 3.16: Creación de un servidor REST

Como se muestra en la siguiente imagen, Hyperledger Composer construye un servidor REST y genera una API con todos los elementos del modelo que puede ser llamada desde servicios externos:

Admin : Rest server methods

org_basic_identificacion_ConfirmarIdentificacion : A transaction named ConfirmarIdentificacion

org_basic_identificacion_Contract : An asset named Contract

org_basic_identificacion_Empleado : A participant named Empleado

org_basic_identificacion_Empresa : A participant named Empresa

org_basic_identificacion_FacialRecognition : A transaction named FacialRecognition

org_basic_identificacion_RechazarIdentificacion : A transaction named RechazarIdentificacion

org_basic_identificacion_Registro : An asset named Registro

org_basic_identificacion_setupProject : A transaction named setupProject

System : General business network methods

Figura 3.20: Hyperledger Composer API-REST

Fuente: Elaboración propia

Dado que, como se detalla en el apartado 3.1.3, existe una limitación técnica al desplegar este servidor en un entorno local y no se puede consumir directamente desde Google Colaboratory, es necesario generar programas desarrollados en Python para interactuar con Hyperledger Fabric desde el entorno local.

En primer lugar, se ha desarrollado un programa para instanciar el modelo y que éste pueda ser puesto en funcionamiento:

```
# Realiza el setup del proyecto haciendo una llamada a la transaccion definida en el modelo de Hyperledger mediante una API-REST
import requests
import base64

# Funcion que realiza la inicializacion del proyecto haciendo la llamada a la API de Hyperledger previamente definida
def setupBlockchain():
  print("Realizando inicializacion del proyecto...")
  resp = requests.post('http://localhost:3000/api/org.basic.identificacion.setupProject')
```

```
  print("Inicializacion realizada")

# Funcion main
def main():
  setupBlockchain()

if __name__ == "__main__":
  main()
```

Listing 3.17: Instancia modelo Hyperledger Fabric

Tras la ejecución de este programa, se puede observar en Hyperledger Composer Playground cómo se han creado todos los elementos que conforman la red:

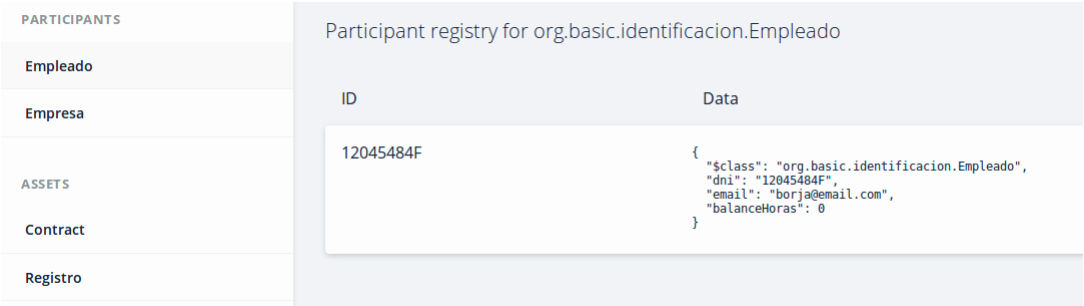


Figura 3.21: Red consultada en Hyperledger Composer Playground

Fuente: Elaboración propia

Por otro lado, se ha desarrollado un programa que lee las trazas registradas por el modelo de identificación facial y las registra en la cadena de bloques, de manera que permanezcan inmutables, que sería el programa principal del sistema:

```
# Realiza el registro de la identificaciones en la cadena de
   bloques a traves de una llamada a la transaccion definida en el
   modelo de Hyperledger mediante una API-REST
import json
import requests
import base64
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive

# Funcion que descarga el fichero de Google Drive
def descargaFichero():
  # Autenticacion en Google Drive
  gauth = GoogleAuth()
  gauth.LocalWebserverAuth()

```

```
14    drive = GoogleDrive(gauth)
15
16    # Descarga del fichero
17    file3 = drive.CreateFile({'id': '1JpLaSM9kSomNcEv6mMuvUXry4yl-
      buqr'})
18    print('Descargando el fichero %s de Google Drive' % file3['title'
      ])
19    file3.GetContentFile('identificacion.json')
20    print('Fichero descargado')
21
22  # Funcion que procesa el fichero descargado
23  def leeFichero():
24    print('Procesando el fichero')
25    with open('identificacion.json') as json_file:
26        data = json.load(json_file)
27    return data
28    print('Fichero procesado')
29
30  # Funcion que escribe en la cadena de bloques el fichero procesado
      a traves de una llamada a la API de Hyperledger definida
31  def escribeBlockchain(data_json):
32    print('Escribiendo en la cadena de bloques...')
33    resp = requests.post('http://localhost:3000/api/org.basic.
      identificacion.FacialRecognition', json=data_json)
34    print('Escritura realizada...')
35
36  # Funcion main
37  def main():
38    descargaFichero()
39    data_json = leeFichero()
40    escribeBlockchain(data_json)
41
42  if __name__ == "__main__":
43    main()
```

Listing 3.18: Registro de las identificaciones en Hyperledger Fabric

```
borja@borja-VirtualBox:~/tfm$ python escribeAPI.py
Your browser has been opened to visit:
()
    https://accounts.google.com/o/oauth2/auth?scope=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fd
rive&redirect_uri=http%3A%2F%2Flocalhost%3A8090%2F&response_type=code&client_id=844511542175-m7
cbtmnjp75ntmpfi0oflojc50hoe9gk.apps.googleusercontent.com&access_type=offline
()
Authentication successful.
Descargando el fichero identificacion.json de Google Drive
Fichero descargado
Procesando el fichero
Escribiendo en la cadena de bloques...
Escritura realizada...
```

Figura 3.22: Escritura de la identificación en Hyperledger Fabric

Fuente: Elaboración propia

Como se puede ver desde Hyperledger Composer Playground, el registro de la traza de identificación se ha realizado de manera correcta en la cadena de bloques.

REG_001


```
{
  "$class": "org.basic.identificacion.Registro",
  "registroId": "REG_001",
  "estado": "PDTE",
  "count": 4,
  "facialrecognitions": [
    {
      "$class": "org.basic.identificacion.FacialRecognition",
      "registro": "resource:org.basic.identificacion.Registro#REG_001",
      "fecha": "2019-08-10T09:38:19.571Z",
      "etiqueta": "Borja",
      "hash_imagen": "fffffff900000000",
      "transactionId": "a5b5e255f522e5b4cf480434181fd1ba9b5b9c08c20755dfd65a9bbe024977ab",
      "timestamp": "2019-08-10T14:28:07.696Z"
    },
    {
      "$class": "org.basic.identificacion.FacialRecognition",
      "registro": "resource:org.basic.identificacion.Registro#REG_001",
      "fecha": "2019-08-10T09:38:19.571Z",
      "etiqueta": "Borja",
      "hash_imagen": "fffffff900000000",
      "transactionId": "ab2f00bc4deda54065295e505d3f2605c2f4449968c46df00a115c881ed2b13f",
      "timestamp": "2019-08-10T14:29:29.131Z"
    },
    {
      "$class": "org.basic.identificacion.FacialRecognition",
      "registro": "resource:org.basic.identificacion.Registro#REG_001",
      "fecha": "2019-08-10T09:38:19.571Z",
      "etiqueta": "Borja",
      "hash_imagen": "fffffff900000000",
      "transactionId": "46fdd88fae29ef45003e2bb6e35bf46229f69dfbc2d46c7288e54963d51243bf",
      "timestamp": "2019-08-10T14:34:03.979Z"
    },
    {
      "$class": "org.basic.identificacion.FacialRecognition",
      "registro": "resource:org.basic.identificacion.Registro#REG_001",
      "fecha": "2019-08-10T09:38:19.571Z",
      "etiqueta": "Borja",
      "hash_imagen": "fffffff900000000",
      "transactionId": "3718bcb2c0e55b4661b81d6775d723c6984685ab4eab5ff0567ce25d2a9c5ea3",
      "timestamp": "2019-08-10T14:37:54.546Z"
    }
  ],
  "contract": "resource:org.basic.identificacion.Contract#CON_002"
}
```


Figura 3.23: Consulta del registro de identificación en Hyperledger Composer Playground

Fuente: Elaboración propia

Así, el sistema desarrollado permite realizar la identificación de individuos mediante técnicas Deep Learning de reconocimiento facial. Además, permite que las identificaciones realizadas se almacenen en una cadena de bloques Hyperledger y permanezcan, de esta manera, inmutables a lo largo del tiempo. Se ha implementa-

do, por tanto, una solución global que, salvando las limitaciones técnicas comentadas, consigue cumplir los objetivos del proyecto.

Capítulo 4

Evaluación y pruebas de la solución

La evaluación del sistema desarrollado será positiva siempre y cuando se satisfagan los objetivos planteados en el capítulo introductorio. Para realizar la evaluación, se ha utilizado un planteamiento basado en la ejecución de casos de prueba que se detallan en los siguientes apartados.

4.1. Proceso de evaluación

La evaluación se ha realizado en base al diseño y ejecución de una serie de casos de prueba que permiten determinar si el sistema desarrollado cumple con los objetivos establecidos. Además, los casos de prueba no sólo evalúan si el sistema es correcto o no, sino que también ofrecen la posibilidad de realizar una detección temprana de errores, permitiendo realizar ajustes durante el desarrollo del proyecto.

4.1.1. Forma de evaluación

Los casos de prueba se pueden definir como un procedimiento de análisis de la calidad del sistema, con el objetivo principal de detectar errores de manera temprana y de aportar información a los interesados en el proyecto.

El momento temporal del desarrollo donde se deben ejecutar los casos de prueba varía en función del objetivo buscado. Algunos casos se pueden plantear desde fases iniciales del desarrollo, como es el caso de las pruebas unitarias o las pruebas de integración. Mientras que las pruebas unitarias verifican el correcto funcionamiento

de los componentes del sistema de manera independiente, las pruebas de integración verifican que la interacción entre los componentes sea correcta. Sin embargo, existen otro tipo de pruebas que requieren que el sistema esté finalizado, para poder ofrecer pruebas automatizadas de principio a fin que simulen un escenario real del sistema desplegado en producción.

Durante el desarrollo del sistema, se han ido realizando pruebas unitarias para verificar que los desarrollos realizados de los componentes eran correctos. A su vez, las pruebas de integración han permitido comprobar que el acoplamiento entre los diferentes componentes desarrollados era correcto y que se cumplía con los requerimientos establecidos.

4.1.2. Casos de prueba y análisis de resultados

En esta sección se detallan los principales casos de prueba realizados, aunque no se especificarán todos y cada uno de ellos para no dilatar de manera excesiva el capítulo. Además, por cada caso de prueba, se muestra si el resultado ha sido el esperado o no.

A la hora de detallar cada caso de prueba se ha seguido el siguiente esquema:

- **Descripción**: descripción del caso de uso.
- **Resultado esperado**: salida esperada tras la ejecución del caso de uso.
- **Resultado obtenido**: salida obtenida tras la ejecución del caso de uso.
- **Evidencia**: muestra de la ejecución de la prueba.
- **Conclusión**: resultado final del caso de uso.

A continuación, se especifican los principales casos de prueba siguiendo el esquema indicado.

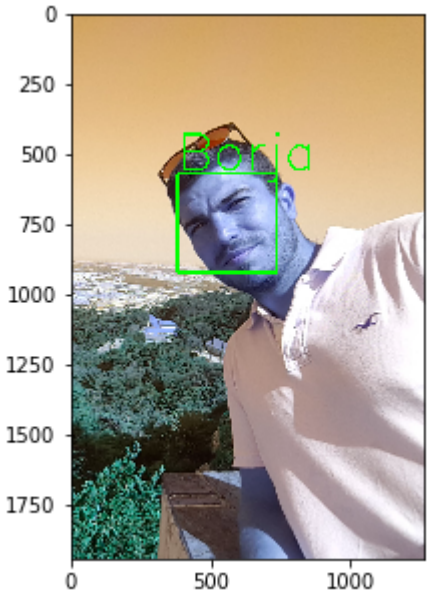
Descripción	Identificación de un individuo mediante el modelo Deep Learning basado en técnicas de reconocimiento facial.
Resultado esperado	Identificación correcta del individuo.
Resultado obtenido	La persona se identifica correctamente.
Evidencia	
Conclusión	Satisfactoria.

Tabla 4.1: Caso de prueba 1

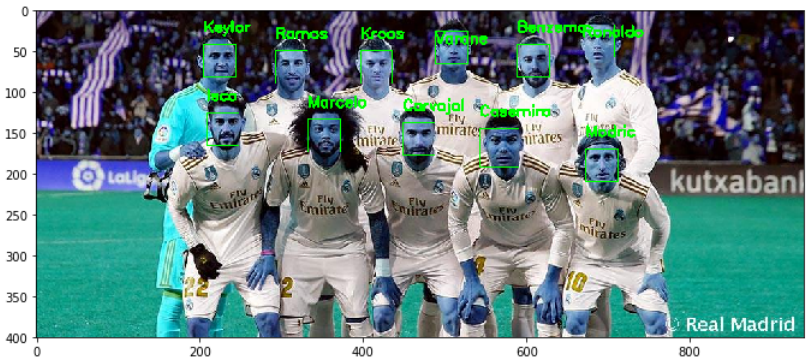
Descripción	Identificación de varios individuos mediante el modelo Deep Learning basado en técnicas de reconocimiento facial.
Resultado esperado	Identificación correcta de los individuos.
Resultado obtenido	Las personas se identifican correctamente.
Evidencia	
Conclusión	Satisfactoria.

Tabla 4.2: Caso de prueba 2

Descripción	Captura de imágenes a través de la cámara web y posterior identificación mediante el modelo Deep Learning.
Resultado esperado	Captura correcta de la imagen.
Resultado obtenido	Imagen capturada correctamente.
Evidencia	
Conclusión	Satisfactoria.

Tabla 4.3: Caso de prueba 3

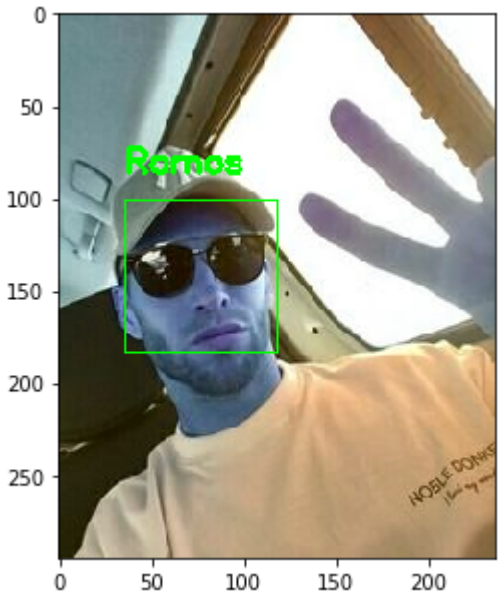
Descripción	Identificación de un individuo con gafas de sol mediante el modelo Deep Learning basado en técnicas de reconocimiento facial.
Resultado esperado	Identificación correcta del individuo.
Resultado obtenido	La persona se identifica correctamente.
Evidencia	
Conclusión	Satisfactoria.

Tabla 4.4: Caso de prueba 4

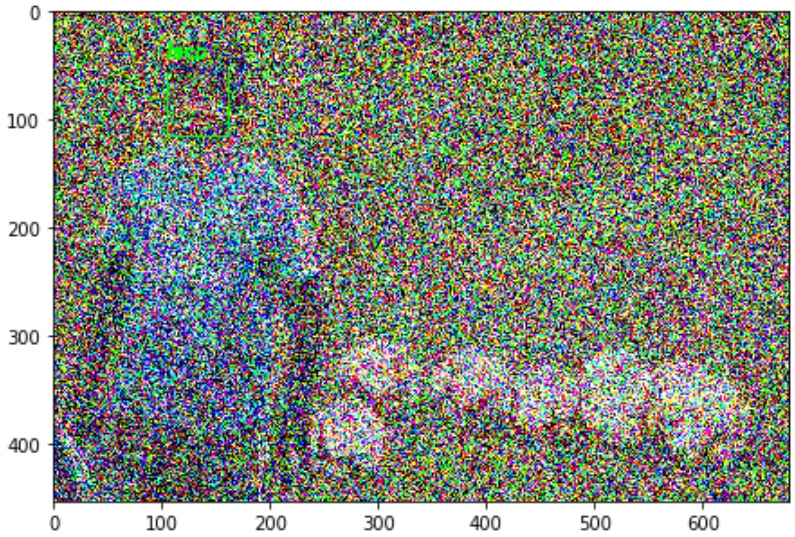
Descripción	Identificación de un individuo mediante el modelo Deep Learning basado en técnicas de reconocimiento facial, añadiendo ruido Gaussiano a las imágenes.
Resultado esperado	Identificación correcta del individuo.
Resultado obtenido	La persona se identifica correctamente.
Evidencia	
Conclusión	Satisfactoria.

Tabla 4.5: Caso de prueba 5

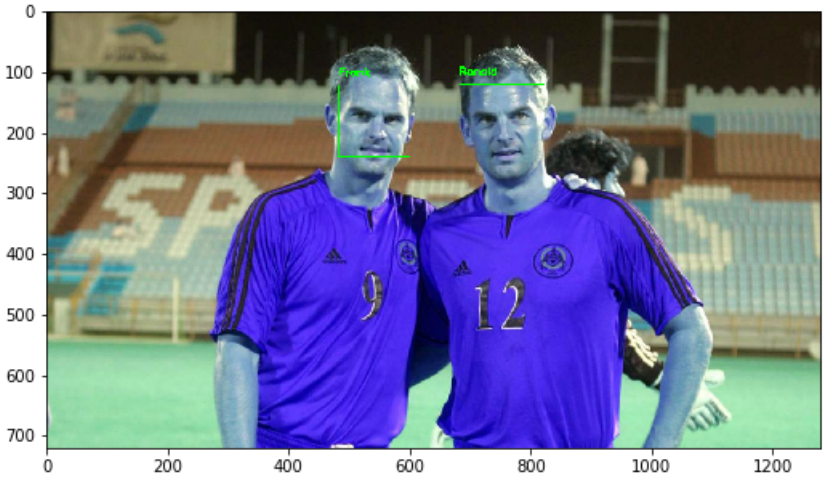
Descripción	Identificación de dos individuos gemelos mediante el modelo Deep Learning basado en técnicas de reconocimiento facial.
Resultado esperado	Identificación correcta de los individuos.
Resultado obtenido	La persona se identifica correctamente.
Evidencia	
Conclusión	Satisfactoria.

Tabla 4.6: Caso de prueba 6

Descripción	Identificación de objetos mediante el modelo Deep Learning basado en técnicas de reconocimiento facial.
Resultado esperado	Los objetos no se identifican utilizando el modelo.
Resultado obtenido	El modelo no está preparado para la identificación de objetos. Es un problema diferente a tratar.
Evidencia	
Conclusión	Satisfactoria.

Tabla 4.7: Caso de prueba 7

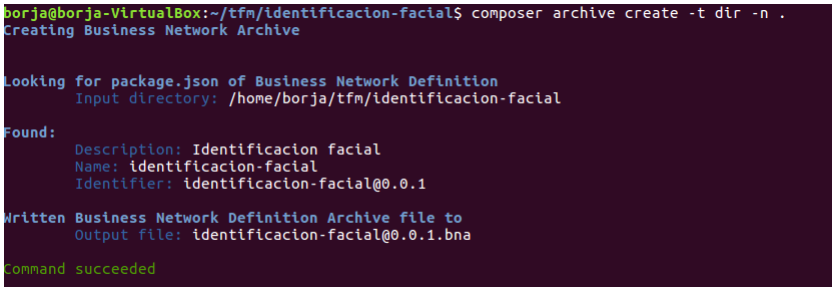
Descripción	Comprobación del despliegue de Hyperledger Fabric.
Resultado esperado	Despliegue correcto de la infraestructura de cadena de bloques.
Resultado obtenido	Hyperledger Fabric se despliega de manera correcta.
Evidencia	
Conclusión	Satisfactoria.

Tabla 4.8: Caso de prueba 8

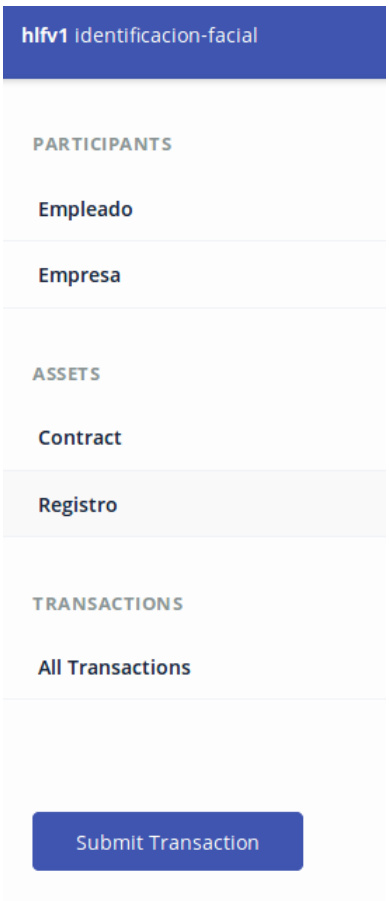
Descripción	Creación del modelo a través de Hyperledger Composer.
Resultado esperado	Despliegue correcto del modelo de cadena de bloques.
Resultado obtenido	El modelo de Hyperledger Composer se despliega de manera correcta.
Evidencia	
Conclusión	Satisfactoria.

Tabla 4.9: Caso de prueba 9

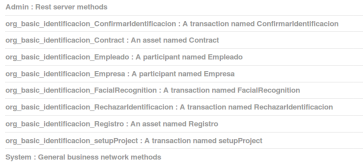
Descripción	Creación del servidor REST de Hyperledger.
Resultado esperado	Despliegue correcto del servidor REST de Hyperledger Composer.
Resultado obtenido	El servidor REST de Hyperledger Composer se despliega de manera correcta.
Evidencia	
Conclusión	Satisfactoria.

Tabla 4.10: Caso de prueba 10

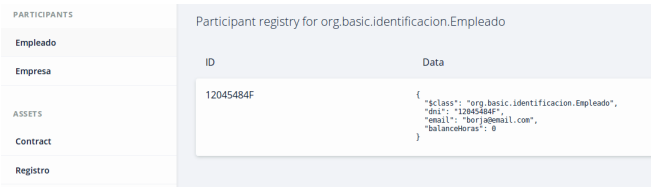
Descripción	Inicialización del proyecto a través de la API desde Python.
Resultado esperado	Inicialización del modelo de Hyperledger Composer desde Python.
Resultado obtenido	Inicialización realizada correctamente.
Evidencia	
Conclusión	Satisfactoria.

Tabla 4.11: Caso de prueba 11

Descripción	Registro de una traza en Hyperledger Fabric.
Resultado esperado	Escritura de un registro de identificación en Hyperledger Fabric desde Python.
Resultado obtenido	Escritura realizada correctamente.

Evidencia	REG_001
	{
	"$class": "org.basic.identificacion.Registro",
	"registroId": "REG_001",
	"estado": "PDTE",
	"count": 4,
	"facialrecognitions": [
	{
	"$class": "org.basic.identificacion.FacialRecognition",
	"registro": "resource:org.basic.identificacion.Registro#REG_001",
	"fecha": "2019-08-10T09:38:19.571Z",
	"etiqueta": "Borja",
	"hash_imagen": "ffffff900000000",
	"transactionId": "a5b5e255f522e5b4cf480434181fd1ba9b5b9c08c20755dfd65a9bbe024977ab",
	"timestamp": "2019-08-10T14:28:07.696Z"
	},
	{
	"$class": "org.basic.identificacion.FacialRecognition",
	"registro": "resource:org.basic.identificacion.Registro#REG_001",
	"fecha": "2019-08-10T09:38:19.571Z",
	"etiqueta": "Borja",
	"hash_imagen": "ffffff900000000",
	"transactionId": "ab2f00bc4deda54065295e505d3f2605c2f4449968c46df00a115c881ed2b13f",
	"timestamp": "2019-08-10T14:29:29.131Z"
	},
	{
	"$class": "org.basic.identificacion.FacialRecognition",
	"registro": "resource:org.basic.identificacion.Registro#REG_001",
	"fecha": "2019-08-10T09:38:19.571Z",
	"etiqueta": "Borja",
	"hash_imagen": "ffffff900000000",
	"transactionId": "46fdd88fae29ef45003e2bb6e35bf46229f69dfbc2d46c7288e54963d51243bf",
	"timestamp": "2019-08-10T14:34:03.979Z"
	},
	{
	"$class": "org.basic.identificacion.FacialRecognition",
	"registro": "resource:org.basic.identificacion.Registro#REG_001",
	"fecha": "2019-08-10T09:38:19.571Z",
	"etiqueta": "Borja",
	"hash_imagen": "ffffff900000000",
	"transactionId": "3718bcb2c0e55b4661b81d6775d723c6984685ab4eab5ff0567ce25d2a9c5ea3",
	"timestamp": "2019-08-10T14:37:54.546Z"
	}
	],
	"contract": "resource:org.basic.identificacion.Contract#CON_002"
	}

Conclusión	Satisfactoria.

Tabla 4.12: Caso de prueba 12

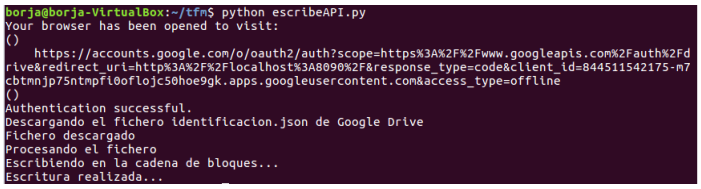
Descripción	Ejecución global del sistema.
Resultado esperado	Ejecución sin errores del sistema global.
Resultado obtenido	El sistema funciona sin ningún error.
Evidencia	 borja@borja-VirtualBox:~/tfm$ python escribeAPI.py Your browser has been opened to visit: () https://accounts.google.com/o/oauth2/auth?scope=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive&redirect_uri=http%3A%2F%2Flocalhost%3A8090%2F&response_type=code&client_id=844511542175-m7cbtmnjp75ntmpfi0oflojc50hoe9gk.apps.googleusercontent.com&access_type=offline () Authentication successful. Descargando el fichero identificacion.json de Google Drive Fichero descargado Procesando el fichero Escribiendo en la cadena de bloques... Escritura realizada...
Conclusión	Satisfactoria.

Tabla 4.13: Caso de prueba 13

Capítulo 5

Líneas futuras

El sistema de identificación desarrollado, aunque es totalmente funcional y es válido como una prueba de concepto, no se puede considerar totalmente finalizado. Bien es cierto que se trata de un desarrollo experimental, con tecnologías en auge actualmente en multitud de sectores.

Dado que, a la hora de la realización del proyecto existía una limitación temporal por la tipología del trabajo, se desarrolló la funcionalidad core del sistema, identificando, a su vez, una serie de líneas de trabajo futuras que permitan perfilar el sistema y dotarlo de más características.

Las líneas de trabajo identificadas son:

- **Integración de la plataforma en un entorno Cloud** que permita el despliegue del sistema bajo la misma infraestructura y evitar, así, las limitaciones técnicas encontradas durante el desarrollo del proyecto.
- **Aplicación del sistema en un ámbito de tiempo real**. Si bien el modelo de detección facial utilizado está preparado para su funcionamiento en tiempo real, sería necesario desarrollar una arquitectura IoT que permitiese la adquisición y el procesamiento de datos en tiempo real.
- **Precisión en imágenes**. Incluir el porcentaje de precisión del modelo de detección facial a la hora de identificar un individuo en una imagen.
- **Ampliación de la funcionalidad del modelo incorporando la detección de objetos**. El modelo utilizado no está pensado para la detección de objetos. No obstante, puede ser interesante incluir esta funcionalidad para apli-

car el sistema en otros problemas (por ejemplo, detección de productos en un supermercado para la compra automática de los mismos).

- **Revisión de la seguridad del sistema desarrollado**. No se ha realizado un análisis de la seguridad global del sistema. En el caso de la integración en un entorno Cloud, habría que establecer una política de seguridad en el mismo.
- **Análisis de la legislación vigente para su cumplimiento**. A la hora de industrializar una solución de este tipo, es necesario revisar todas la normativa vigente dado que, al tratar con información personal y confidencial, es una problemática muy sensible, siendo de vital importancia GDPR (General Data Protection Regulation).
- **Mejora de la red de negocio**. Las reglas establecidas en la red de negocio implementada son muy básicas. Mediante la mejora de estas reglas, se podría dotar al sistema de mayor funcionalidad (por ejemplo, realizando un registro automático de las horas extras de un empleado o registrando los retrasos a la entrada del mismo).
- **Notificaciones en el sistema**. Incluir un sistema de notificaciones que avise a un individuo cada vez que ha sido identificado, permitiéndole confirmar o denegar su identidad.
- **Implementación de interfaz gráfica**. Desarrollar una interfaz gráfica que permita consultar la información almacenada en la cadena de bloques y que sustituya el entorno web Playground de Hyperledger Composer.

Capítulo 6

Conclusiones

El desarrollo de soluciones innovadoras, que proporcionan nuevas e interesantes funcionalidades a la sociedad, lleva asociado la necesidad de evolucionar y digitalizar muchos procesos actuales con el objetivo de sustentar las necesidades reales de las personas bajo la aplicación de las nuevas tecnologías. A su vez, las personas requieren cada vez un grado de interacción menor con determinadas tareas, solicitando la automatización de las mismas.

En este sentido, las técnicas de Deep Learning permiten realizar la automatización y resolución de muchos problemas, siendo en la mayoría de los casos, más precisas y eficientes que los seres humanos.

Por otro lado, cada vez se requiere que la privacidad a la hora de realizar cualquier tarea sea mayor. Además, se ve incrementada la importancia de la trazabilidad de la información, requiriendo, en muchos casos, una prueba sobre la creación o la existencia de la información en un marco temporal muy amplio. Es bajo este escenario, donde la aplicación de la tecnología Blockchain proporciona grandes ventajas, favoreciendo la creación de infraestructuras más seguras y trazables que no requieren la intervención de terceros para proporcionar confianza a un sistema.

Ante estas dos situaciones, se ha desarrollado un sistema de identificación basado en técnicas de detección facial, proyecto que realiza una prueba de concepto sencilla, clara y que demuestra cómo se pueden complementar e integrar dos tecnologías en auge en la actualidad. El sistema propuesto trata de demostrar las ventajas de la integración de ambas tecnologías bajo la definición de un caso de uso claro y funcional.

Por último, es importante indicar que el desarrollo de este proyecto no pretende únicamente ser el trabajo final de un máster, sino que tiene un ciclo de vida mayor, pues se tiene pensado continuar con el desarrollo del mismo, abordando las líneas futuras indicadas en el capítulo anterior y, así, poder presentarlo en algún congreso y/o publicarlo en alguna revista científica.

Bibliografía

[1] World Bank. *Identification for Development (ID4D)*. URL: https://id4d.worldbank.org/.

[2] United Nations. *Sustainable Development Goal (SDG)*. URL: https://www.un.org/sustainabledevelopment/sustainable-development-goals/.

[3] Reham Amin, Tarek Gaber, Ghada Eltaweel y Aboul Ella Hassanien. «Biometric and Traditional Mobile Authentication Techniques: Overviews and Open Issues». En: *Intelligent Systems Reference Library* (2014).

[4] Cadena SER. *Robo de identidad*. URL: https://cadenaser.com/emisora/2019/02/22/ser_madrid_sur/1550844120_215839.html.

[5] Ioannis Tzemos, Apostolos P. Fournaris y Nicolas Sklavos. «Security and Efficiency Analysis of One Time Password Techniques». En: *Proceedings of the 20th Pan-Hellenic Conference on Informatics*. 2016.

[6] Lin Hong y Anil Jain. «Integrating faces and fingerprints for personal identification». En: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1998).

[7] Hazem El-Bakry y Nikos Mastorakis. «Personal identification through biometric technology». En: *Proceedings of the 9th WSEAS International Conference on APPLIED INFORMATICS AND COMMUNICATIONS (AIC '09)* (2009).

[8] Rakesh Yadhav y Choodarathnakara. «An Efficient Approach for Robust Fingerprint Recognition System». En: *2015 International Journal of Scientific Research in Science* (2015).

[9] Ayon Dey. *Machine Learning Algorithms: A Review*. 2016.

[10] Sitio Big Data. *Machine Learning: Seleccion Métricas de clasificación*. URL: http://sitiobigdata.com/2019/01/19/machine-learning-metrica-clasificacion-parte-3/.

[11] Corinna Cortes y Vladimir Vapnik. «Support vector networks». En: *Machine Learning* (1995).

[12] Lidnsay I Smith. *A tutorial on Principal Components Analysis*. 2002.

[13] S. Setiowati, Zulfanahri, E. L. Franita e I. Ardiyanto. «A review of optimization method in face recognition: Comparison deep learning and non-deep learning methods». En: *2017 9th International Conference on Information Technology and Electrical Engineering (ICITEE)*. 2017.

[14] B. Poon, M. A. Amin y Hong Yan. «PCA Based face recognition and testing criteria». En: *2009 International Conference on Machine Learning and Cybernetics* (2009).

[15] M. M. Sani, K. A. Ishak y S. A. Samad. «Evaluation of Face Recognition System Using Support Vector Machine». En: *2009 IEEE Student Conference on Research and Development (SCOReD)* (2009).

[16] UCSD Computer Vision. *Yale Face Database*. URL: `http://vision.ucsd.edu/content/yale-face-database`.

[17] Ian Goodfellow, Yoshua Bengio y Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016.

[18] Rina Dechter. «Learning While Searching in Constraint-satisfaction-problems». En: *Proceedings of the Fifth AAAI National Conference on Artificial Intelligence*. 1986.

[19] Nur Hordri, Siti Yuhaniz y Siti Mariyam Shamsuddin. *Deep Learning and Its Applications: A Review*. 2016.

[20] Linnea Claesson y Magnus Jendbro. *Deep learning methods and applications - Classification of traffic signs and detection of Alzheimer's disease from images*. 2017.

[21] Kevin Gurney. *An Introduction to Neural Networks*. Taylor Francis, Inc., 1997.

[22] Bohdan Macukow. «Neural Networks – State of Art, Brief History, Basic Models and Architecture». En: *15th IFIP International Conference on Computer Information Systems and Industrial Management (CISIM)*. Springer International Publishing, 2016.

[23] MathWorks. *Deep Learning: Tres cosas que es necesario saber*. URL: `https://es.mathworks.com/discovery/deep-learning.html`.

[24] Li Deng. «Three Classes of Deep Learning Architectures and Their Applications: A Tutorial Survey». En: *APSIPA-2011 Conference*. 2011.

[25] Mei Wang y Weihong Deng. «Deep Face Recognition: A Survey». En: *CoRR* (2018).

[26] Alex Krizhevsky, Ilya Sutskever y Geoffrey E. Hinton. «ImageNet Classification with Deep Convolutional Neural Networks». En: *Neural Information Processing Systems* (2012).

[27] Karen Simonyan y Andrew Zisserman. «Very Deep Convolutional Networks for Large-Scale Image Recognition». En: *arXiv 1409.1556* (2014).

[28] Y. Taigman, M. Yang, M. Ranzato y L. Wolf. «DeepFace: Closing the Gap to Human-Level Performance in Face Verification». En: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014.

[29] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke y A. Rabinovich. «Going deeper with convolutions». En: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015.

[30] F. Schroff, D. Kalenichenko y J. Philbin. «FaceNet: A unified embedding for face recognition and clustering». En: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015.

[31] K. He, X. Zhang, S. Ren y J. Sun. «Deep Residual Learning for Image Recognition». En: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[32] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj y L. Song. «SphereFace: Deep Hypersphere Embedding for Face Recognition». En: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[33] J. Hu, L. Shen y G. Sun. «Squeeze-and-Excitation Networks». En: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018.

[34] Stephen Balaban. «Deep learning and face recognition: the state of the art». En: *arXiv:1902.03524* (2015).

[35] University of Massachusetts. *Labeled Faces in the Wild*. URL: `http://vis-www.cs.umass.edu/lfw/`.

[36] Jesús Iglesias García. «Desarrollo de un sistema de trazabilidad en entornos IoT mediante Hyperledger». Tesis doct. Universidad Autónoma de Madrid, 2018.

[37] Roger Wattenhofer.

[38] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. 2009. URL: `http://www.bitcoin.org/bitcoin.pdf`.

[39] Leslie Lamport, Robert Shostak y Marshall Pease. «The Byzantine Generals Problem». En: *ACM Transactions on Programming Languages and Systems (TOPLAS)* (1982).

[40] Cristina Pérez Solà y Jordi Herrera Joancomartí. *Bitcoins y el problema de los generales bizantinos*. 2014.

[41] S. Zhang y J. Lee. «Double-spending with a Sybil Attack in the Bitcoin Decentralized Network». En: *IEEE Transactions on Industrial Informatics* (2019).

[42] Markus Jakobsson y Ari Juels. «Proofs of Work and Bread Pudding Protocols». En: *Proceedings of the IFIP TC6/TC11 Joint Working Conference on Secure Information Networks: Communications and Multimedia Security*. 1999.

[43] Scott Nadal Sunny King. *PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake*. 2012.

[44] Jesús Díaz David Arroyo y Luis Hernández. *Blockchain (Qué sabemos de)*. Consejo Superior de Investigaciones Cientificas, 2019.

[45] Vitalik Buterin. *Ethereum White Paper*. 2013. URL: `http://blockchainlab.com/pdf/Ethereum_white_paper-a_next_generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf`.

[46] Maher Alharby, Amjad Aldweesh y Aad van Moorsel. «Blockchain-based Smart Contracts: A Systematic Mapping Study of Academic Research». En: *arXiv:1710.06372* (2018).

[47] Ivica Nikolic, Aashish Kolluri, Ilya Sergey, Prateek Saxena y Aquinas Hobor. «Finding The Greedy, Prodigal, and Suicidal Contracts at Scale». En: *Proceedings of the 34th Annual Computer Security Applications Conference*. 2018.

[48] Deloitte. *Deloitte's 2019 Global Blockchain Survey*. URL: `https://www2.deloitte.com/content/dam/Deloitte/se/Documents/risk/DI_2019-global-blockchain-survey.pdf`.

[49] Charlie Lee. *Litecoin*. URL: `https://litecoin.org/es/..`

[50] Evan Duffield y Daniel Díaz. *Dash: A Privacy-Centric Crypto-Currency*. 2015. URL: `https://docisolation.prod.fire.glass/?guid=9b76d741-42e7-4977-ab32-b285882e26ba`.

[51] R3. *R3: Enterprise Blockchain Platform*. URL: `https://www.r3.com/`.

[52] Hyperledger. *Hyperledger*. URL: `https://www.hyperledger.org/`.

[53] Enterprise Ethereum Alliance. *Enterprise Ethereum Alliance*. URL: `https://entethalliance.org/`.

[54] Post Trade Distributed Ledger. *Post Trade Distributed Ledger*. URL: `http://www.ptdlgroup.org/`.

[55] IBM. *IBM Pricing*. URL: `https://www.ibm.com/es-es/cloud/pricing`.

[56] Project Management Institute. *A Guide To The Project Management Body Of Knowledge (PMBOK Guides)*. Project Management Institute, 2004.

[57] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad y Michael Stal. *Pattern-Oriented Software Architecture - Volume 1: A System of Patterns*. Wiley Publishing, 1996.

[58] K. He, X. Zhang, S. Ren y J. Sun. «Deep Residual Learning for Image Recognition». En: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[59] Davis King. *Dlib Models*. URL: https://github.com/davisking/dlib-models.

[60] Borja Simancas Ruesgas. *Repositorio del proyecto*. URL: https://github.com/bsr92/TFM.git.

[61] The Hacker Factor Blog. *Average hash*. URL: http://www.hackerfactor.com/blog/index.php?/archives/432-Looks-Like-It.html.

[62] Johannes Buchner. *ImageHash*. URL: https://pypi.org/project/ImageHash/.

[63] Muntasir Mamun. *How does Hyperledger Fabric works*. URL: https://medium.com/coinmonks/how-does-hyperledger-fabric-works-cdb68e6066f5.