



UNIVERSIDAD DE EXTREMADURA

CENTRO UNIVERSITARIO DE MÉRIDA

GRADO EN INGENIERÍA INFORMÁTICA EN TECNOLOGÍAS DE LA INFORMACIÓN

TRABAJO FIN DE GRADO

**CONTROL DE ASISTENCIA A CLASE MEDIANTE UN LECTOR
DE HUELLA DACTILAR**

AUTOR: Víctor Manuel Calle Sánchez

Mérida, Noviembre de 2016



UNIVERSIDAD DE EXTREMADURA

CENTRO UNIVERSITARIO DE MÉRIDA

GRADO EN INGENIERÍA INFORMÁTICA EN TECNOLOGÍAS DE LA INFORMACIÓN

TRABAJO FIN DE GRADO

**CONTROL DE ASISTENCIA A CLASE MEDIANTE UN LECTOR
DE HUELLA DACTILAR**

AUTOR: Víctor Manuel Calle Sánchez

Fdo.:

DIRECTOR: Héctor Sánchez Santamaría

Fdo.:

Agradecimientos

Lo primero de todo es agradecer al director de mi Trabajo Fin de Grado, Héctor Sánchez Santamaría, quien tuvo la idea de realizar este proyecto y me metió de lleno en algo sobre lo que no esperaba que iba a ser tan interesante. También agradecer su apoyo, correcciones, consejos y ayuda durante todo el proceso de realización del mismo.

Por otro lado me gustaría agradecer a todos mis familiares que me inculcaron el trabajo, estudio y aprendizaje como principios de vida desde pequeño, y que a la postre me apoyaron durante los cuatro años de Universidad para finalmente tener este título.

También doy las gracias a todos mis amigos que me rodearon durante mis estudios en Mérida, en especial a Alejandro Castillo, Gabriel Cano, Francisco Ramírez y Esteban Pérez, por acompañarme en estos cuatro años y por habernos ayudado el uno al otro siempre que lo necesitábamos.

En definitiva, a todos los nombrados y a la comunidad educativa del Centro Universitario de Mérida, muchas gracias.

Resumen

El Plan Bolonia se presentó como una renovación del mundo universitario adaptado a la empresa y al marco europeo de integración de todos los estudiantes de la comunidad.

Dentro de este marco de renovación se incluyeron algunas novedades a las que el entorno universitario ha tenido que adaptarse. Entre ellas, el Plan Bolonia incluye el control de asistencia a clase y tutoría del profesorado y de asistencia a clase para el alumnado.

Esto obliga a los centros a equiparse con un control de asistencia fiable, seguro y a su vez, sencillo de manejar. De esta manera, el control de asistencia del profesorado se va asegurar mediante el uso de la biometría, más concretamente de un lector de huella digital conectado a un ordenador.

En este Trabajo Fin de Grado se ha diseñado y desarrollado un sistema informático que permite, a través de un sensor de huellas dactilares, registrar la entrada y salida del profesor a sus clases y a las tutorías oficiales validadas por el departamento.

Los fichajes se pueden llevar a cabo a través de una aplicación web simple y corporativa, y estos quedan registrados en un servidor de base de datos. Además, se pueden generar diferentes informes que recojan las asistencias o faltas a cada clase y tutoría de cada uno de los profesores.

Palabras clave: biometría, lector de huellas, control de asistencia.

Abstract

The Bologna Process is presented as a renewal of the academic world adapted to the enterprise and to the European framework of integration of all college students.

In this framework of renewal is included some novelties which the university has had to adapt. For example, the Bologna Process includes the class and tutorship of faculty attendance control and the class attendance for student body.

So the university centers are obligated to set up an assistance control reliable, safe and easy to handle. In this way, the attendance control for faculty it will ensure with biometrics, specifically with a digital fingerprint reader connecting to a laptop.

With this project, we have designed and developed a computer system that permits through a fingerprint sensor, to register the input and output of the teacher their classes and the official validated tutorships from the department.

The signings can be done through a corporate and simple web application, and they are stored in a database server. Moreover, we can generate different reports which inform the attendance or absence of each class and tutorship from faculty.

Keywords: biometrics, fingerprint sensor, attendance control.

x

Índice

Capítulo 1: Introducción	1
1.1. Introducción	1
1.2. Antecedentes	2
1.3. Objetivos	3
Capítulo 2: Análisis previo	4
2.1. Sistemas Biométricos	5
2.1.1. Biometría.....	5
2.1.2. Arquitectura de los sistemas biométricos	7
2.1.3. Características biométricas.....	10
2.1.4. Tipos de sistemas biométricos: rasgos físicos o estáticos	10
2.1.5. Tipos de sistemas biométricos: rasgos dinámicos	15
2.1.6. Tabla comparativa de los sistemas biométricos	18
2.1.7. Usos y aplicaciones de la biometría	19
2.2. Huella dactilar	21
2.2.1. Fundamentos de las huellas dactilares	21
2.2.2. Procesamiento de la huella digital.....	25
2.3. Tipos y funcionamiento de los distintos lectores de huella dactilar	26
2.3.1. Lectores ópticos	27
2.3.2. Lectores de capacitancia	28
2.3.3. Lectores mecánicos	29
2.3.4. Lectores térmicos.....	29
2.3.5. Lectores LE (<i>Light Emitting</i>)	29
2.3.6. Lectores de salida dinámica.....	30
2.4. Ejemplos de sistemas de huella dactilar	30
2.4.1. DigitalPersona U.are.U	30
2.4.2. Lumidigm.....	31
2.4.3. Nitgen Fingkey Hamster	32
2.4.4. ATMEL Fingerchip	33

2.4.5. Fulcrum Biometrics	33
2.4.6. UPEK TouchChip.....	34
2.4.7. Lector de huellas GT-511C3	35
2.4.8. Sensor de huellas ADAFRUIT con Arduino.....	36
2.5. Tipos de SDKs para el desarrollo de aplicaciones basadas en sistemas de huella dactilar	37
Capítulo 3: Metodología de Desarrollo.....	40
3.1. Introducción	40
3.1.1. Etapas del modelo iterativo e incremental	41
3.1.2. Ventajas y desventajas del modelo iterativo e incremental.....	42
Capítulo 4: Análisis del Sistema.....	44
4.1. Planificación del sistema	44
4.1.1. Viabilidad Técnica	44
4.1.2. Viabilidad Operacional	45
4.1.3. Viabilidad Económica	45
4.1.4. Viabilidad de la Solución	46
4.2. Análisis funcional.....	46
4.3. Diagramas de casos de usos detallados	47
4.3.1. Casos de uso detallados del actor Profesor	48
4.3.2. Casos de uso detallados del actor Administrador.....	51
4.4. Diagramas de actividades	53
4.5. Cronograma.....	54
Capítulo 5: Diseño del Sistema.....	56
5.1. Arquitectura del sistema	56
5.2. Diseño de la instalación Hardware	58
5.2.1. Arduino	58
5.2.2. Lector de huellas.....	59
5.2.3. Pantalla LCD.....	59
5.2.4. Servidor Raspberry PI 2	60
5.3. Patrón MVC en nuestro proyecto	60
5.3.1. Capa de Presentación.....	61
5.3.2. Capa Lógica	62
5.3.3. Capa del Modelo	62
5.4. Sistema de reservas: MRBS	62

5.4.1. Modelo de datos MRBS.....	63
5.4.2. Ampliación y variaciones en el modelo de datos.....	64
5.5. Diseño de la seguridad.....	68
5.5.1. Seguridad en la identificación biométrica	68
5.5.2. Seguridad en el servidor y en la aplicación web	68
5.5.3. Encriptación AES	69
Capítulo 6: Implementación.....	70
6.1. Análisis y diseño de la arquitectura	70
6.1.1. Capa de Presentación.....	70
6.1.2. Capa lógica.....	73
6.1.3. Capa del modelo	84
6.2. Arquitectura Hardware	86
6.2.1. Programa Arduino.....	87
6.2.2. Servidor Raspberry Pi 2	89
6.3. Implementación de la seguridad	90
6.3.1. Implementación de seguridad en la identificación biométrica.....	90
6.3.2. Implementación de seguridad en el servidor y en la aplicación web....	90
Capítulo 7: Conclusiones y trabajos futuros.....	91
7.1. Conclusiones.....	91
7.2. Trabajos futuros	92
7.3. Planificación estimada y planificación real	94
Capítulo 8: Bibliografía	96
Capítulo 9: Anexos.....	103
9.1. ANEXO I: Instalación de Apache, MySQL y PHP	104
9.1.1. Introducción	104
9.1.2. Apache	104
9.1.3. PHP.....	105
9.1.4. MYSQL y PHPMyAdmin	105
9.2. ANEXO II: Instalación de MRBS.....	108
9.2.1. Introducción	108
9.2.2. Añadir gestor de tutorías	109
9.2.3. Establecer una dirección IP fija	110
9.3. ANEXO III: Arduino y lector de huellas Adafruit	112

9.3.1.	Introducción	112
9.3.2.	Instalación de Arduino IDE	112
9.3.3.	Instalación de la librería Adafruit	112
9.3.4.	Conexiones entre el Arduino, el lector de huellas y la pantalla LCD ...	113
9.3.5.	Conexión Java - Arduino.....	114
9.3.6.	Puerto de conexión Arduino.....	115
9.4.	ANEXO IV: Jasper Reports	117
9.4.1.	Introducción	117
9.4.2.	Instalación de Jasper Reports en Eclipse	117
9.4.3.	Creación de informes con Jasper Reports	118
9.5.	ANEXO V: Despliegue de la aplicación web.....	124
9.5.1.	Introducción	124
9.5.2.	Instalación de XAMPP	124

Índice de figuras

Figura 1: tipos de sistemas biométricos	6
Figura 2: proceso de inscripción	8
Figura 3: proceso de identificación de usuario	8
Figura 4: proceso de verificación de usuario	9
Figura 5: sistema biométrico por emisión de calor	11
Figura 6: sistema biométrico por huella digital	11
Figura 7: sistema biométrico por composición química del olor corporal	12
Figura 8: sistema biométrico por escáner de retina	12
Figura 9: sistema biométrico por escáner de iris	13
Figura 10: sistema biométrico por geometría de la mano	14
Figura 11: sistema biométrico por características faciales	15
Figura 12: sistema biométrico por el movimiento corporal.....	15
Figura 13: sistema biométrico por verificación de voz.....	16
Figura 14: sistema biométrico por la dinámica de tecleo. A la izquierda, un gráfico de ejemplo comparando diferentes tiempos de tecleo entre 14 usuarios.....	17
Figura 15: sistema biométrico por firma manuscrita	17
Figura 16: porcentaje de tecnologías biométricas más utilizadas. Análisis realizapor el International Biometric Group	21
Figura 17: gráfico de una huella donde podemos ver el sistema dactilar marginal (A), nuclear (B) y basilar (C).....	22
Figura 18: gráfico de una huella donde se indica un núcleo (core) y un delta	23
Figura 19: huella dactilar con arcos	23
Figura 20: huella dactilar con presillas internas	23
Figura 21: huella dactilar con presillas externas	24
Figura 22: huella dactilar con verticilos.....	24
Figura 23: ejemplos de minucias	24
Figura 24: tipos de huellas: lazo derecho (a), lazo izquierdo (b), circular (c), arco (d) y arco con tendencia (e).....	25
Figura 25: resumen de los pasos necesarios para el procesamiento de una huella	27
Figura 26: corrección de la orientación de la huella para lectores ópticos	28
Figura 27: proceso de reconocimiento de una huella en lectores de capacitancia	28
Figura 28: Digital U.are.U Persona 4500	31
Figura 29: Lumidigm Venus Series sensor	32
Figura 30: NITGEN Fingkey Hamster.....	32
Figura 31: ATMEL Fingerchip	33
Figura 32: Fulcrum Biometrics FbF MobileOne	34
Figura 33: UPEK TouchChip TCRU1C	35
Figura 34: Fingerprint Scanner GT-511C3	35

Figura 35: Adafruit Fingerprint Sensor	36
Figura 36: Modelo de ciclo de vida iterativo e incremental.....	41
Figura 37: etapas del modelo iterativo e incremental	42
Figura 38: diagrama de casos de uso.....	47
Figura 39: diagrama de actividades	54
Figura 40: registro de huella [45]	57
Figura 41: verificación de huella.....	57
Figura 42: patrón Modelo-Vista-Controlador [47]	58
Figura 43: ejemplo de un sensor ADAFRUIT conectado a una base shield Grove de Arduino	59
Figura 44: diseño de la aplicación bajo el patrón MVC	61
Figura 45: diagrama de la base de datos (parte 1).....	66
Figura 46: diagrama de la base de datos (parte 2).....	67
Figura 47: imagen de nuestro sistema de conexión Arduino.....	86
Figura 48: imagen de nuestra Raspberry Pi 2 conectada a Internet por cable RJ-45, a la corriente a través de un Micro-USB, y a una pantalla por medio del HDMI. Cuenta con una tarjeta SD de 32 GB.....	89
Figura 49: el comando “service apache2 status” nos muestra que Apache está activo	105
Figura 50: el comando “service mysql status” nos muestra que MySQL está activo ..	106
Figura 51: las aplicaciones web se despliegan en la carpeta html de www (Apache) .	106
Figura 52: funcionamiento de la página test.php en el servidor Apache	107
Figura 53: PHPMyAdmin.....	107
Figura 54: aplicación MRBS	109
Figura 55: aplicación MRBS con gestor de Tutorías	109
Figura 56: configuración NAT para dirección IP fija	110
Figura 57: creación de dominio en NO-IP.....	110
Figura 58: privilegios del usuario mrbs_remoto (permite conexiones exteriores).....	111
Figura 59: context.xml que conecta con la base de datos MRBS de la Raspberry Pi 2	111
Figura 60: ejemplos de la librería del sensor de huellas Adafruit	113
Figura 61: sensor de huellas conectado a una placa Grove, que a su vez se integra en un Arduino UNO	114
Figura 62: pantalla Grove-LCD RGB Backlight	114
Figura 63: puerto Arduino en Windows	115
Figura 64: puerto Arduino en plataforma Linux.....	116
Figura 65: JasperSoft Studio	117
Figura 66: creación del adaptador.....	118
Figura 67: configuración de la conexión JDBC en el adaptador	118
Figura 68: creación de fichero Jasper Report en Eclipse	119
Figura 69: selección de la base de datos e introducción de la consulta que realizará el informe	119
Figura 70: selección de los campos recuperados por la consulta	120
Figura 71: editor del informe Jasper Report	121
Figura 72: incorporación de parámetros en una consulta	122

Figura 73: instalador de XAMPP	124
Figura 74: fichero de configuración tomcat-users.xml.....	125
Figura 75: interfaz gráfica de XAMPP	126
Figura 76: Apache Tomcat.....	126
Figura 77: realización de despliegues en Apache Tomcat.....	127
Figura 78: lista de aplicaciones desplegadas en Apache	127

Índice de tablas

Tabla 1: comparativa de las características generales de los diferentes sistemas biométricos (1)	18
Tabla 2: comparativa de las características generales de los diferentes sistemas biométricos (2)	19
Tabla 3: características DigitalPersona 4500	31
Tabla 4: características Sensor Venus	31
Tabla 5: características Fingkey Hamster II DX.....	32
Tabla 6: características ATMEL Fingerchip	33
Tabla 7: características Fulcrum Biometrics FbF MobileOne	34
Tabla 8: características UPEK TouchChip TCRU1C.....	34
Tabla 9: características Fingerprint Scanner GT-511C3.....	35
Tabla 10: características Adafruit Fingerprint Sensor	36
Tabla 11: características del Biokey SDK	37
Tabla 12: características del SDK DigitalPersona Pro Enterprise	37
Tabla 13: características del VeriFinger SDK	38
Tabla 14: características del IDKit SDK	38
Tabla 15: características del Free Fingerprint Verification SDK	38
Tabla 16: características del BioMini SDK.....	39
Tabla 17: características del software necesario para los lectores de huellas con plataforma Arduino	39
Tabla 18: cuantía total de los dispositivos necesarios	45
Tabla 20: coste total del desarrollo de las aplicaciones	45
Tabla 21: caso de uso detallado "Registrar huella"	48
Tabla 22: caso de uso detallado "Fichaje en clase"	48
Tabla 23: caso de uso detallado "Fichaje en tutorías"	49
Tabla 24: caso de uso detallado "Consultar fichajes"	49
Tabla 25: caso de uso detallado "Justificar una falta"	50
Tabla 26: caso de uso detallado "Registrar tutorías"	50
Tabla 27: caso de uso detallado "Solicitar reserva"	50
Tabla 28: caso de uso detallado "Consultar fichajes"	51
Tabla 29: caso de uso detallado "Generar informes"	51
Tabla 30: caso de uso detallado "Configurar lector de huellas"	52
Tabla 31: caso de uso detallado "Eliminar huella"	52
Tabla 32: caso de uso detallado "Registrar usuario".....	53
Tabla 33: planificación estimada	55
Tabla 34: planificación real.....	94

Índice de algoritmos

Algoritmo 1: ejemplo de código en una página JSP donde se refleja el uso de JSP, HTML y Bootstrap	72
Algoritmo 2: código del servlet Action	74
Algoritmo 3: ejemplo de código en el AdminController (método para el informe de aula por días)	75
Algoritmo 4: ejemplo de código en el HuellaController (método para eliminar una huella)	76
Algoritmo 5: ejemplo de código en el InformeController (método para el reporte de tutorías de un profesor)	76
Algoritmo 6: ejemplo de código en el ProfesorController (método para mostrar todas sus tutorías)	77
Algoritmo 7: ejemplo de código en el UsuarioController (método con el que se valida a un usuario que intenta acceder con sus credenciales)	78
Algoritmo 8: ejemplo de código en el ClaveService (método para guardar una huella)	79
Algoritmo 9: ejemplo de código en el FichajeService (método para mostrar la lista de fichajes completa)	80
Algoritmo 10: ejemplo de código en el InformeServic (método para obtener las titulaciones)	81
Algoritmo 11: ejemplo de código en el ReservaService (método para buscar una reserva)	82
Algoritmo 12: ejemplo de código en el TutoriaService (método para buscar una tutoría)	83
Algoritmo 13: ejemplo de código en el UsuarioService (método para obtener el identificador de un usuario)	84
Algoritmo 14: código para modificar la tabla mrbs_fichaje	85
Algoritmo 15: código para crear la tabla mrbs_claves	86
Algoritmo 16: código para crear la tabla mrbs_titulacion	86
Algoritmo 17: código para modificar la tabla mrbs_asigprof	86
Algoritmo 18: ejemplo de código en el programa Arduino (método para obtener una coincidencia de la huella)	89
Algoritmo 19: ejemplo de código en un JSP que embebe a un informe de JasperReport	123

Capítulo 1: Introducción

Contenidos

Capítulo 1: Introducción	1
1.1. Introducción	1
1.2. Antecedentes	2
1.3. Objetivos	3

En este primer capítulo se va a introducir este trabajo fin de grado. Para ello, se abordarán los siguientes aspectos: introducción, antecedentes y objetivos del proyecto.

1.1. Introducción

Según el artículo 164 de las normativas y estatutos internos de la Universidad de Extremadura [1], es un deber del Personal Docente e Investigador de la Universidad de Extremadura:

“Cumplir fielmente sus obligaciones docentes, investigadoras o de otra índole, con el alcance y dedicación que se establezcan para cada categoría, manteniendo actualizados sus conocimientos y de acuerdo con las normas deontológicas y éticas que correspondan”.

Además, el artículo 16 también nos dice que son funciones de las Facultades y Escuelas *“la coordinación y supervisión de su actividad docente”*.

Esto obliga a los centros a equiparse con un control de asistencia fiable, seguro, económico y a su vez, sencillo de manejar que permita adquirir información acerca de la asistencia de profesorado. Para ello, sería imprescindible un sistema de acceso que aporte la seguridad necesaria para tomar dicho control, como puede ser la biometría. Un ejemplo serían las huellas dactilares, un sistema rápido, cómodo y preciso.

El funcionamiento es muy sencillo y seguro. El profesor antes de comenzar la clase o en algún momento de la jornada de celebración de tutorías, se dirige a la conserjería. Allí encontrará un ordenador portátil con un lector de huella digital, donde realizará el fichaje y verificará su asistencia. Los fichajes quedan registrados en un programa de gestión de asistencia a través del cual se podrán generar los informes requeridos por el Servicio de Inspección.

En este Trabajo Fin de Grado pretendemos diseñar y desarrollar un sistema informático que permita, a través de un lector de huella digital, registrar la entrada y salida del profesor a sus clases y las tutorías oficiales validadas por el departamento. Además, el programa de gestión de asistencia permitiría generar informes o estadísticas.

1.2. Antecedentes

Hasta el momento, el control de asistencia a clase y a tutoría por parte del profesorado se ha llevado a cabo en el Centro Universitario de Mérida mediante hojas de firmas: el profesor llega y se pasa por conserjería para recoger las herramientas que necesite para impartir la clase, y realiza una firma manuscrita en el apartado de su asignatura que está contenido en la hoja de firmas.

En lo que respecta a las tutorías, el Servicio de Inspección recomienda / exige que desde la dirección de los centros / facultades se haga un seguimiento de cumplimiento por parte del profesorado de sus tutorías.

Este método presenta una serie de problemas importantes que ya se analizaron en el Trabajo Fin de Grado de Milagros Membrillo Jiménez llamado “**eCUM: control de asistencia**” (desconocimiento de la hora exacta de llegada, proceso rudimentario de búsqueda entre los papeles, así como el gasto de papel y energético para las impresiones...). En la presente se persigue controlar la asistencia del profesorado mediante tarjetas NFC que sirven para realizar el fichaje a través de una aplicación móvil, sustituyendo así las hojas de firmas. El dispositivo se encontraría en conserjería y el profesor, al coger las llaves y demás material para el aula, efectuaría su fichaje en ese momento.

Esto presenta también varios inconvenientes, los cuales pretendemos solventar en el desarrollo de este Trabajo Fin de Grado:

- Necesidad de portar una **tarjeta NFC** cada vez que se asista a clase. Este método es propenso a olvidos o pérdidas de la tarjeta por parte del usuario.
- **Posibilidad de que alguien fiche por otro.** Una tarjeta NFC es un objeto físico que como tal puede ser prestado a otra persona para que ésta realice el fichaje en su lugar. Mediante huella dactilar, una huella es única y no puede ser suplantada por otra persona.
- Tiene que haber un **dispositivo móvil** en conserjería para realizar el fichaje, y, además, debe disponer de un lector NFC.
- **Menor nivel de seguridad.** Una tarjeta NFC puede ser perfectamente robada, mientras que para registrar la asistencia por biometría con un lector de huellas, solo es necesario pasar nuestro propio dedo a través del sensor.
- **Problemas de compatibilidad:** no todos los dispositivos móviles tienen la ventaja de disponer de un lector NFC.

- **Fichaje más lento y más incómodo.** Tener que conectarse al dispositivo móvil y pasar la tarjeta NFC a la hora de fichar resulta un método más incómodo y más lento que el uso de un lector de huella: se llega al sensor y se apoya el dedo.

1.3. Objetivos

Los objetivos marcados para la realización de este trabajo Fin de Grado han sido:

- **Estudiar y analizar los sistemas hardware de lectura de huella dactilar.** Se realizará un análisis detallado acerca de la biometría y los distintos sistemas biométricos, profundizando en los diferentes tipos de lectores de huella digital y su proceso para conseguir la identificación del individuo.
- **Estudiar y analizar los distintos SDK libres y de pago para el desarrollo de aplicaciones basadas en la lectura de huella dactilar.** También se analizará los diferentes kits de desarrollo de software (SDK) con los cuales se pueden desarrollar aplicaciones con la función de aprovechar totalmente las funcionalidades que ofrecen los sensores de huellas.
- **Diseño y desarrollo de sistema de fichaje de entrada y salida a clase y a tutorías del profesor.** Se desarrollarán las aplicaciones necesarias para que el profesor pueda realizar el fichaje de su asistencia tanto de entrada como de salida a clase y a tutorías.
 - **Aplicación web para registrar la huella.** Se tendrá una aplicación a través de la cual primeramente almacenaremos la huella de cada profesor en la base de datos y en el lector, para que así posteriormente puedan ser identificados cuando realicen un fichaje.
 - **Aplicación web para fichar entrada y salida a clase y a tutorías.** El fichaje se llevará a cabo a través de una aplicación la cual desplegará un sistema de reservas una vez que haya sido identificado el profesor por medio de su huella dactilar, previamente almacenada.

Capítulo 2: Análisis previo

Contenidos

Capítulo 2: Análisis previo	4
 2.1. Sistemas Biométricos.....	5
 2.1.1. Biometría.....	5
 2.1.2. Arquitectura de los sistemas biométricos	7
 2.1.3. Características biométricas.....	10
 2.1.4. Tipos de sistemas biométricos: rasgos físicos o estáticos	10
 2.1.5. Tipos de sistemas biométricos: rasgos dinámicos	15
 2.1.6. Tabla comparativa de los sistemas biométricos	18
 2.1.7. Usos y aplicaciones de la biometría	19
 2.2. Huella dactilar	21
 2.2.1. Fundamentos de las huellas dactilares	21
 2.2.2. Procesamiento de la huella digital.....	25
 2.3. Tipos y funcionamiento de los distintos lectores de huella dactilar	26
 2.3.1. Lectores ópticos	27
 2.3.2. Lectores de capacitancia	28
 2.3.3. Lectores mecánicos	29
 2.3.4. Lectores térmicos.....	29
 2.3.5. Lectores LE (<i>Light Emitting</i>)	29
 2.3.6. Lectores de salida dinámica.....	30
 2.4. Ejemplos de sistemas de huella dactilar	30
 2.4.1. DigitalPersona U.are.U.....	30
 2.4.2. Lumidigm.....	31
 2.4.3. Nitgen Fingkey Hamster	32
 2.4.4. ATMEL Fingerchip	33
 2.4.5. Fulcrum Biometrics	33
 2.4.6. UPEK TouchChip.....	34
 2.4.7. Lector de huellas GT-511C3	35
 2.4.8. Sensor de huellas ADAFRUIT con Arduino.....	36

2.5. Tipos de SDKs para el desarrollo de aplicaciones basadas en sistemas de huella dactilar **37**

En este capítulo se realizará un análisis de las diferentes soluciones de identificación que existen en la actualidad a través de la biometría, así como los fundamentos y el proceso que se lleva a cabo en el uso de la huella dactilar como solución biométrica. El capítulo se divide en los siguientes apartados:

1. **Sistemas biométricos.** Este apartado de iniciación tratará de explicar la biometría, su arquitectura y los diferentes tipos de sistemas biométricos que existen.
2. **Huella dactilar.** Este apartado se divide en dos:
 - a. **Fundamentos de las huellas dactilares.** Esta sección describe los conceptos fundamentales para comprender el uso y juego que dan las huellas dactilares como sistema de reconocimiento único.
 - b. **Procesamiento de la huella digital.** Tras analizar las huellas dactilares como tal, pasamos a ver brevemente los pasos que se siguen a la hora de procesar una imagen de una huella digital.
3. **Tipos y funcionamiento de los distintos lectores de huella dactilar.** En este punto se incluye una explicación sobre seis tipos diferentes que existen actualmente para leer huellas dactilares, profundizando quizás más en los lectores de tipo óptico y de capacitancia.
4. **Ejemplos de sistemas de huella dactilar.** Una vez analizado los tipos de lectores de huellas, vemos en el quinto apartado ejemplos de diferentes sistemas con los que podemos realizar la lectura de huellas dactilares.
5. **Tipos de SDKs para el desarrollo de aplicaciones basadas en sistemas de huella dactilar.** Como para usar un sistema lector de huella se requiere la utilización de un software que permita la interactividad con el hardware, este apartado analiza con unas tablas comparativas un conjunto de las mejores tecnologías SDK útiles para el reconocimiento de huella digital.

2.1. Sistemas Biométricos

2.1.1. Biometría

El concepto de biometría proviene del griego “*bios*” que significa vida, y “*metron*”, cuyo significado es medida. Así, podemos decir que la **biometría** es el estudio automático para el reconocimiento de humanos basados en uno o más rasgos de conducta o físicos intrínsecos [2].

El concepto como tal se define como “*la parte de la biología que estudia en forma cuantitativa la variabilidad individual de los seres vivos por medio de métodos estadísticos*”. Al automatizar este estudio utilizando métodos matemáticos y ayudados por ordenadores, se pasa a llamar **Biometría informática** [3].

Para ello, estas técnicas matemáticas y estadísticas se aplican sobre rasgos de un individuo que son características morfológicas únicas y que lo diferencian para su autenticación, es decir, verificar su identidad. Estas características físicas o estáticas y de comportamiento o dinámicas conforman los diferentes tipos de **sistemas biométricos**.

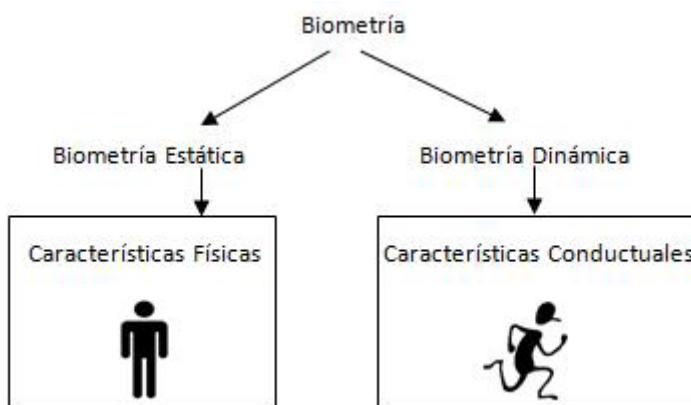


Figura 1: tipos de sistemas biométricos

En definitiva, la medición biométrica se ha venido estudiando con anterioridad y es considerada actualmente como el método ideal de identificación humana y control de acceso. Esta identificación consiste en comparar esas características específicas de cada persona con un patrón conocido y almacenado en una base de datos. Estos sistemas automatizados que realizan tareas de biometría son los sistemas biométricos [4].

Ventajas

En el ámbito de la informática doméstica o laboral, la tecnología biométrica permite que el usuario se identifique en diferentes plataformas sin necesidad de recordar contraseñas o llevar una tarjeta o llave para acceder a un edificio, por ejemplo, lo que aumenta la comodidad [5].

Son sistemas muy robustos y proporcionan un control de acceso físico fiable en entornos que requieren alta seguridad, puesto que la biometría garantiza uno de los niveles de autenticación menos franqueables en la actualidad. Además, las características biométricas de una persona son intransferibles a otra.

Por otro lado, la utilización de un dispositivo biométrico permite que los costos de administración sean más pequeños, ya que sólo se debe realizar el mantenimiento del lector, y que una persona se encargue de mantener la base de datos actualizada [2].

Inconvenientes

Algunos sistemas biométricos utilizan tecnologías muy caras, y son difíciles de implantar.

Según un estudio de la Comisión Europea realizado en 2005 [5], no es una tecnología que sea compatible 100% con toda la población, ya que se estima que un 5% de personas con minusvalías no pueden utilizar este tipo de sistemas de seguridad.

Además, como todo sistema informático, puede ser susceptible a errores en la identificación, a ataques de sabotaje o a que pueda ser burlado mediante suplantación de identidad, aunque esto ocurre de manera puntual, como por ejemplo, cuando alguien consigue la propia huella del dueño.

El principal problema podría ser que los rasgos biométricos están considerados como datos de carácter personal y por tanto su tratamiento por parte de empresas e instituciones está regulado por la actual ley de Protección de Datos [5].

2.1.2. Arquitectura de los sistemas biométricos

La arquitectura típica de un sistema biométrico puede entenderse conceptualmente como dos módulos [6]:

- **Módulo de inscripción (*enrollment module*)**. Se encarga de adquirir y almacenar la información proveniente del indicador biométrico para poder contrastar a ésta con la proporcionada en ingresos posteriores al sistema. Las labores ejecutadas por el módulo de inscripción son posibles gracias a la acción del lector biométrico y del extracto de características.

En cuanto al proceso del módulo de inscripción, los dispositivos biométricos poseen tres componentes básicos:

- El primero se encarga de la **adquisición análoga o digital de algún indicador biométrico de una persona**, como por ejemplo, la adquisición de la imagen de una huella dactilar mediante un escáner.
- El segundo maneja la **compresión, procesamiento, almacenamiento y comparación de los datos adquiridos** (en el ejemplo, a partir de la salida del lector, extraemos una imagen) **con los datos almacenados**.
 - **Template**. Un template es la información representativa del indicador biométrico que se encuentra almacenada y será utilizada en los labores de identificación al ser comparada con la información proveniente del indicador biométrico en el punto de acceso.
- El tercer componente establece una **interfaz con aplicaciones** ubicadas en el mismo u otro sistema.

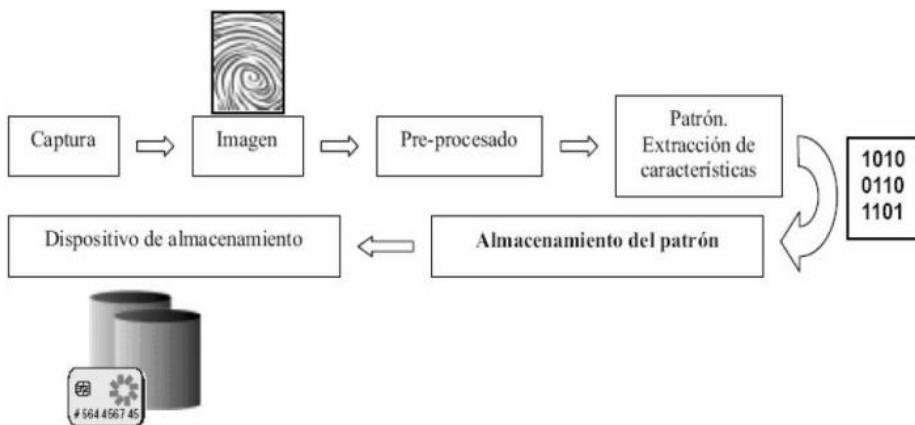


Figura 2: proceso de inscripción

- **Módulo de identificación (*identification module*)**. Se encarga de reconocer a los individuos en la aplicación de control de acceso. El proceso de identificación se inicia cuando el lector biométrico captura la característica única del individuo y la convierte a formato digital, para que el extracto de características produzca una representación compacta con igual formato que los **templates**. La representación consultante, llamada **query**, se envía al comparador de características que confronta a éste con uno o varios *templates* para establecer la identidad.
 - **Fase operacional.** Los procesos realizados por el módulo de identificación.

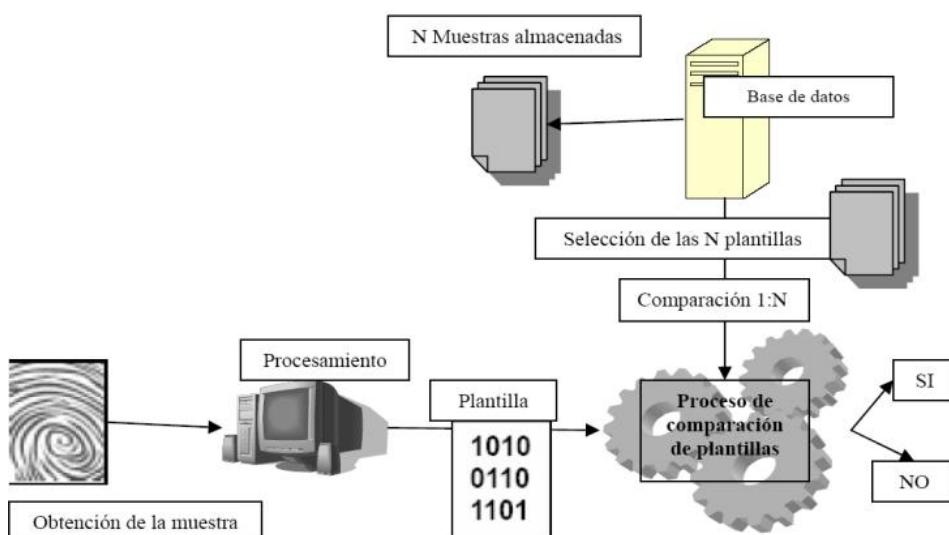


Figura 3: proceso de identificación de usuario

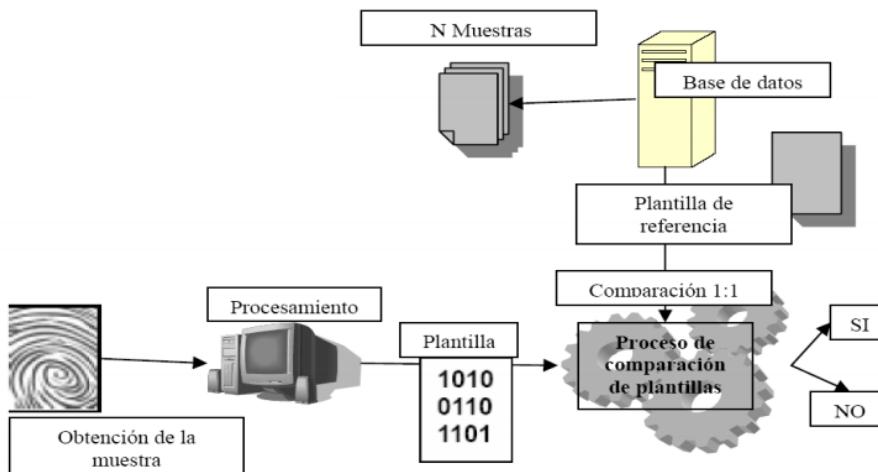


Figura 4: proceso de verificación de usuario

La información provista por los **templates** permite particionar su base de datos de acuerdo a la presencia o no de ciertos patrones particulares para cada indicador biométrico. Las "clases" así generadas permiten reducir el rango de búsqueda de algún template en la base de datos. Sin embargo, los templates pertenecientes a una misma clase también presentarán diferencias conocidas como **variaciones intraclasses**. Éstas implican que la identidad de una persona puede ser establecida sólo con un cierto **nivel de confianza** [7].

Una decisión tomada por un sistema biométrico distingue "**personal autorizado**" o "**impostor**". Para cada tipo de decisión, existen dos posibles salidas, verdadero o falso. Por lo tanto existe un total de cuatro posibles respuestas del sistema [8] [9]:

- Una persona autorizada es aceptada (salida correcta)
- Una persona autorizada es rechazada (salida incorrecta)
- Un impostor es rechazado (salida correcta)
- Un impostor es aceptado (salida incorrecta)

El grado de confidencia asociado a las diferentes decisiones puede ser caracterizado por la distribución estadística del número de personas autorizadas e impostores. Estas estadísticas se utilizan para establecer dos tasas de errores:

- **Tasa de falsa aceptación (FAR: False Acceptance Rate)**. Es la frecuencia relativa con que un impostor es aceptado como un individuo autorizado.
- **Tasa de falso rechazo (FRR: False Rejection Rate)**. Es la frecuencia relativa con que un individuo autorizado es rechazado como un impostor.

La FAR y la FRR son **funciones del grado de seguridad** deseado. Normalmente el resultado del proceso de identificación o verificación será un número real normalizado en el intervalo $[0, 1]$, que indicará el "**grado de parentesco**" o correlación entre la característica biométrica proporcionada por el usuario y la almacenada en la base de datos. **Una FRR pequeña usualmente entrega una FAR alta, y viceversa.**

2.1.3. Características biométricas

Las características básicas que un sistema biométrico para identificación personal debe cumplir pueden expresarse mediante las restricciones que deben ser satisfechas. Estas restricciones apuntan a que el sistema considere [7] [9] [10]:

- **El desempeño**, que se refiere a la exactitud, el rendimiento, la rapidez y la robustez alcanzada en la identificación. El objetivo de esta restricción es comprobar si el sistema posee una exactitud y rapidez aceptable con un requerimiento de recursos razonable.
- **La aceptabilidad por parte del usuario**, que indica el grado en que la gente está dispuesta a aceptar un sistema biométrico en su vida diaria. Es claro que el sistema no debe representar peligro alguno para los usuarios y debe inspirar "confianza" a los mismos.
- **La fiabilidad**, que refleja cuán difícil es burlar al sistema, cómo de resistente es ante un posible fraude o usurpación. El sistema biométrico debe reconocer características de una persona viva, pues es posible crear dedos de látex, grabaciones digitales de voz, prótesis de ojos, etc. En definitiva, esta característica mide el nivel de seguridad que satisface el sistema biométrico.
- **Universalidad**: todos los usuarios tienen que tener la característica que solicita el sistema biométrico en cuestión.
- **Singularidad o univocidad**. La característica a medir tiene que tener carácter distintivo: *la huella dactilar de una persona es única y ninguna otra persona tendrá la misma*.
- **Permanencia** en el tiempo y ante condiciones ambientales diversas.
- **Colectividad**. La característica ha de ser mensurable cuantitativamente.

2.1.4. Tipos de sistemas biométricos: rasgos físicos o estáticos

La biometría estática se basa en medidas o datos de partes del cuerpo humano, es decir, rasgos físicos del usuario [4] [7] [9] [11]. Ejemplos:

- **Emisión de calor**. Mide la emisión de calor del cuerpo (**termograma**) y realiza un mapa de valores sobre la forma de cada persona.

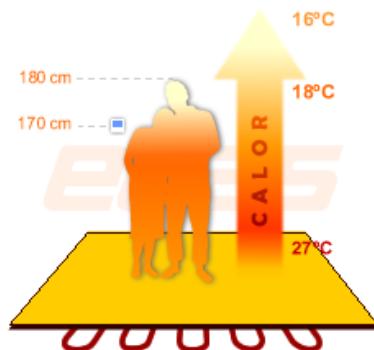


Figura 5: sistema biométrico por emisión de calor

- **Huella digital.** Sigue el principio de que no existen dos huellas dactilares exactamente iguales.

Es un buen sistema con excelentes resultados: cada huella digital tiene un conjunto de pequeños arcos, ángulos, bucles, remolinos, etc. llamados **minucias**. La posición relativa de cada una de ellas es lo que se analiza para establecer la identificación de una persona. Está aceptado que dos personas no tienen más de ocho minucias iguales y cada una posee más de 30, lo que hace al método muy seguro.



Figura 6: sistema biométrico por huella digital

- **Composición química del olor corporal.** Investigadores de la Universidad Politécnica de Madrid, junto con la empresa Ilía Sistemas, están avanzando en el estudio de una nueva técnica biométrica que permite identificar a las personas a partir de su olor corporal. Los resultados de esta investigación revelan que existen patrones reconocibles en el olor corporal de cada individuo que se mantienen constantes. Así, cada persona tiene un olor característico que permite identificarla dentro de un grupo de individuos con una tasa de acierto superior al 85% [12].



Figura 7: sistema biométrico por composición química del olor corporal

- **Verificación de patrones oculares.** Son sistemas que están basados en patrones del iris o de la retina y hasta el momento son considerados los más efectivos ya que en 200 millones de personas la probabilidad de coincidencia es casi de 0. Hay dos formas de escanear los ojos:
 - **Escáner de retina:** mide el patrón de venas en el fondo del ojo. Se obtiene proyectando una luz infrarroja de baja intensidad en forma de espiral, a través de la pupila. Se detectan los nodos y ramas del área retinal para compararlos con los almacenados en una base de datos.
 - **El proceso de adquisición de la imagen de la retina** es el paso más difícil en cualquier sistema de identificación basado en la misma, pues es un órgano interno (es la capa más interna de las tres capas del globo ocular), pequeño y difícil de medir si se carece de un dispositivo apropiado especialmente diseñado para esta aplicación.
 - **El usuario debe situarse muy cerca del dispositivo de captura.** La imagen se captura a través de la pupila y éste debe permanecer inmóvil durante la captura de la imagen, ya que cualquier pequeño movimiento invalidaría el proceso de adquisición de la imagen y se tendría que empezar de nuevo.



Figura 8: sistema biométrico por escáner de retina

- **Escáner de iris:** se realiza utilizando una videocámara con un objetivo de aproximación, para enfocar en el ojo a una distancia del sujeto que no le resulte incómoda, y examinando los patrones de color únicos de los surcos de la parte coloreada de nuestros ojos.
 - Una de las características que hacen del iris una aplicación potencial para la identificación biométrica son su **estabilidad frente a los cambios originados por accidentes**, esto debido a la protección que le confiere la córnea.
 - **La detección del fraude** (por ejemplo si se presenta una fotografía o un ojo de plástico, u otro material, con el iris pintado) se puede realizar de forma sencilla capturando dos fotogramas consecutivos de la imagen, y comparar la dilatación de la pupila, que debe ser distinta. También se pueden forzar cambios controlados de la iluminación para analizar la respuesta de la pupila a dichos cambios.
 - **La principal desventaja** de esta metodología es que es posible que las personas se nieguen a que les analicen los ojos, ya que a través del iris se pueden detectar y/o predecir enfermedades que puede que se prefiera mantener en secreto.



Figura 9: sistema biométrico por escáner de iris

- **Geometría de la mano.** A diferencia de las huellas dactilares, la mano humana no es única, y sus características individuales no son suficientes para identificar a una persona. Sin embargo, su perfil resulta útil si el sistema biométrico lo combina con imágenes individuales de algunos dedos, extrayendo datos como las *longitudes, anchuras, alturas, posiciones relativas, articulaciones...*
 - Estas características se transforman en una **serie de patrones numéricos** que pueden ser comparados. Su principal aplicación es la verificación de usuario.

- **Estos sistemas son los más rápidos dentro de los biométricos:** con una probabilidad de error aceptable, en aproximadamente un segundo son capaces de determinar si una persona es quien dice ser.
- **El funcionamiento de un sistema biométrico de reconocimiento de la mano** es el siguiente:
 1. Cuando un usuario necesita ser autenticado sitúa su mano sobre un dispositivo lector con unas guías que marcan la posición correcta para la lectura.
 2. Una vez la mano está correctamente situada, unas cámaras toman una imagen superior y otra lateral, de las que se extraen ciertos datos en un formato de tres dimensiones.
 3. Transformando estos datos en un modelo matemático que se contrasta contra una base de patrones, el sistema es capaz de permitir o denegar acceso a cada usuario.
- **Uno de los elementos más importantes del reconocimiento mediante la geometría de la mano es que éstos son capaces de aprender:** a la vez que autentican a un usuario, actualizan su base de datos con los cambios que se puedan producir en la muestra (un pequeño crecimiento, adelgazamiento, el proceso de cicatrizado de una herida...).



Figura 10: sistema biométrico por geometría de la mano

- **Características faciales.** Un sistema de reconocimiento facial es una aplicación asistida por ordenador para identificar automáticamente a una persona en una imagen digital mediante la comparación de determinadas características faciales. Aunque estos sistemas requieren alta capacidad de almacenamiento y se precisan imágenes de buena calidad.
 - El método más común utiliza una **cámara para capturar una imagen de nuestra cara**, que es analizada en función de ciertos “**puntos clave**”, como la distancia entre los ojos o la anchura de la nariz.
 - Para el reconocimiento facial existen dos tipos: 2D y 3D.
 1. Algunos **métodos 2D** son los siguientes: redes neuronales, análisis de la geometría facial, comparación de grafos, autocaras, **fisherface** (utiliza información intra-clase para maximizar la separación entre clases).
 2. Los **sistemas 3D** aumentan la fiabilidad del sistema eliminando algunos problemas como la iluminación y las distintas poses, y tienen suficiente información invariante frente a cambios de

expresión, uso de gafas, etc. Algunos ejemplos son: comparación de ejemplos, modelos 3D basados en firmas de puntos, comparación mediante segmentación de la superficie, **AURA** (Advanced Uncertain Reasoning Architecture).

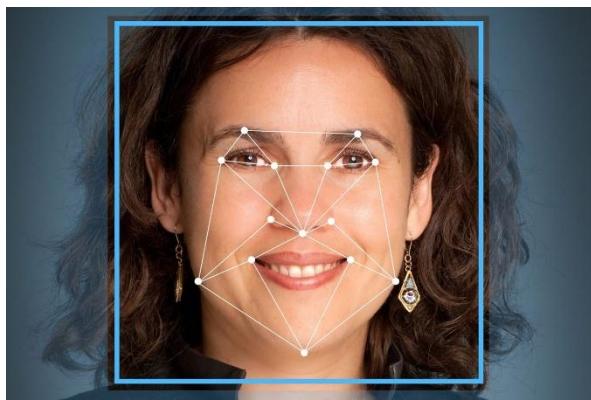


Figura 11: sistema biométrico por características faciales

2.1.5. Tipos de sistemas biométricos: rasgos comportamentales o dinámicos

Por otro lado, la biometría dinámica se basa en las medidas o datos de acciones de una persona, e indirectamente en sus características físicas. Atienden a la conducta o comportamiento del usuario. Ejemplos [3] [4] [7] [9]:

- **Gesto y movimiento corporal.** La expresión corporal de cada persona varía de unas a otras. Un claro ejemplo de un gesto característico entre diferentes usuarios es la manera de andar:

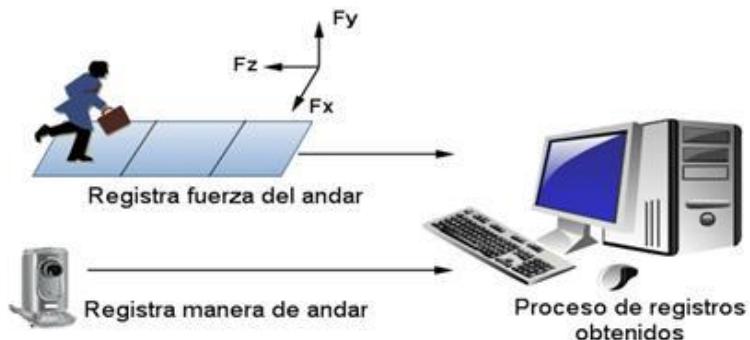


Figura 12: sistema biométrico por el movimiento corporal

- **Verificación de voz.** Consiste en grabar la lectura de una o varias frases por parte de los diferentes usuarios. En el momento de intentar acceder, se comparan la voz con sus diferentes cualidades (entonación, timbre, etc.). Esta tecnología tiene tres formas de reconocer la voz:
 - **Dependencia:** siempre se repite el mismo texto.
 - **Texto aleatorio:** el sistema indica un texto aleatorio a repetir.
 - **Independencia de texto:** el usuario puede decir lo que quiera.

Este sistema es **muy sensible a factores externos** como el ruido, el estado de ánimo y enfermedades que alteren la voz. Además, la variabilidad presente en la señal de la voz cuando se realiza el proceso de identificación resulta perjudicial, pues el locutor no puede repetir de forma exacta una misma frase o palabra.

El reconocimiento de voz generalmente consta de los tres pasos siguientes:

- **Preprocesamiento.** Los sonidos consisten en cambios de presión del aire a través del tiempo y a frecuencias que podemos escuchar. Estos sonidos pueden ser digitalizados por un micrófono o cualquier otro medio que convierte la presión del aire en pulsos eléctricos. En el procesamiento de la señal se extraen las características que utilizará posteriormente el reconocedor.
- **Reconocimiento.** En la etapa de reconocimiento se traduce la señal de entrada a su texto correspondiente. Este proceso se puede llevar a cabo de diversas formas utilizando enfoques como Redes Neuronales Artificiales (RNA) y Modelos Ocultos de Markov (HMM), entre otros.
- **Comunicación.** El resultado de la etapa de reconocimiento será enviado al sistema que lo requiere.

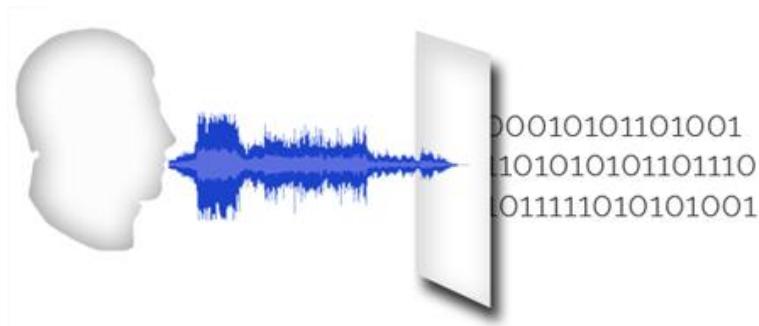


Figura 13: sistema biométrico por verificación de voz

- **Dinámica del tecleo.** El principal mecanismo de interacción de una persona con un ordenador es el teclado. Uno de los dispositivos de comportamiento biométrico es el análisis “**keystroke**”, también llamado “**typing biometrics**”. Este último comportamiento biométrico se refiere a la velocidad con que un individuo emplea el teclado para introducir su identificación o clave de acceso, lo cual puede ser indicativo de la autenticidad del usuario.

En el momento de la captura de la muestra, se tendrá en cuenta:

- **Tiempo entre pulsaciones (latencias):** se mide el intervalo entre la pulsación de una tecla y la siguiente, dentro de una determinada secuencia de tecleo.
- **Tiempo de pulsaciones (duraciones):** en una pulsación específica, se mide cuánto tiempo se mantiene presionada una tecla.
- Una vez obtenidas la latencia y la duración en el tecleo, se hace un **patrón estadístico** y se determina una firma de tecleo para cada usuario.

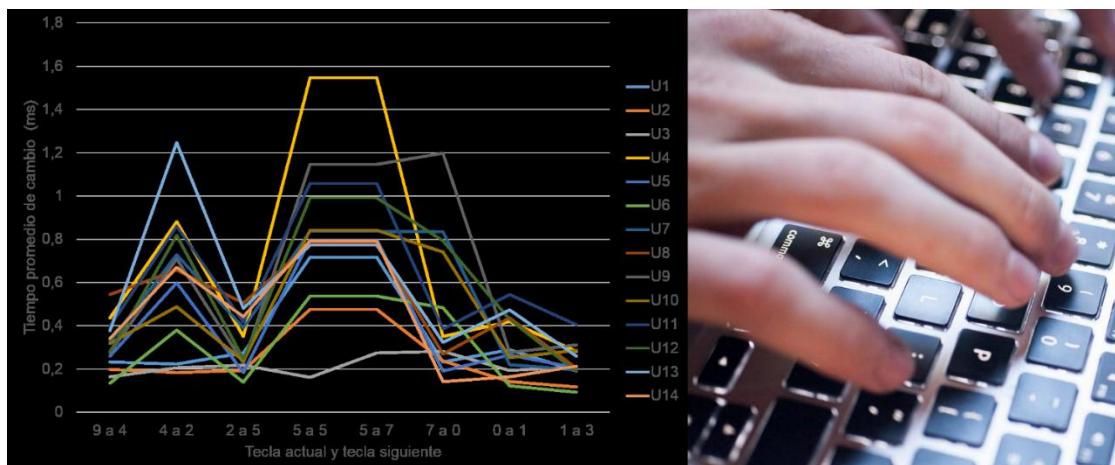


Figura 14: sistema biométrico por la dinámica de tecleo. A la izquierda, un gráfico de ejemplo comparando diferentes tiempos de tecleo entre 14 usuarios

- **Firma manuscrita.** La verificación en base a firmas es algo que todos utilizamos y aceptamos día a día. No obstante, existe una diferencia fundamental entre el uso de las firmas que hacemos en nuestra vida cotidiana y los sistemas biométricos: mientras que habitualmente la verificación de la firma consiste en un simple análisis visual sobre una impresión en papel (**estática** o **reconocimiento off-line**), en los sistemas automáticos no es posible autenticar usuarios en base a la representación de los trazos de su firma. En los modelos biométricos se utiliza además de la forma de firmar, las características **dinámicas** (*Dynamic Signature Verification, reconocimiento de firma on-line*): el tiempo utilizado para firmar, las veces que se separa el bolígrafo del papel, el ángulo con que se realiza cada trazo...
 - Para utilizar un sistema de autenticación basado en firmas se solicita en primer lugar a los futuros usuarios un **número determinado de firmas ejemplo**, de las cuales el sistema extrae y almacena ciertas características. Ante las diferentes firmas del mismo usuario, la solución es relajar las restricciones del sistema a la hora de aprender firmas, con lo que se reduce su seguridad.
 - La firma se captura con un **lápiz óptico** o con una **lectora sensible**.



Figura 15: sistema biométrico por firma manuscrita

2.1.6. Tabla comparativa de los sistemas biométricos

Aquí podemos ver un resumen de las características generales de cada sistema biométrico visto anteriormente [2] [4]:

	Iris	Retina	Huella dactilar	Geometría de la mano
Fiabilidad	Muy alta	Muy alta	Alta	Alta
Facilidad de uso	Media	Baja	Alta	Alta
Prevención de ataques	Muy alta	Muy alta	Alta	Alta
Aceptación	Media	Media	Media	Alta
Estabilidad	Alta	Alta	Alta	Media
Identificación y autenticación	Ambas	Ambas	Ambas	Autenticación
Interferencias	Gafas, iluminación	Irritaciones, gafas, lentillas	Suciedad, heridas, asperezas	Artritis, reumatismo, edad, lesiones
Coste	Alto	Alto	Bajo	Alto
Nivel de seguridad	Alto	Alto	Alto	Medio
Ratio de error	1/131000	1/10 ⁶	1/500+	1/500

Tabla 1: comparativa de las características generales de los diferentes sistemas biométricos (1)

	Escritura y firma	Voz	Patrones faciales
Fiabilidad	Alta	Alta	2D: Media 3D: Alta
Facilidad de uso	Alta	Alta	Alta
Prevención de ataques	Media	Media	2D: Media 3D: Alta
Aceptación	Muy alta	Alta	Muy alta
Estabilidad	Media	Media	2D: Media 3D: Alta
Identificación y autenticación	Ambas	Autenticación	Autenticación
Interferencias	Firmas fáciles o cambiantes, lesiones de mano	Ruido, enfermedades	Pelo, gafas, edad, iluminación
Coste	Bajo	Bajo	Bajo
Nivel de seguridad	Medio	Medio	Medio
Ratio de error	Firma: 1/50	1/50	Sin datos

Tabla 2: comparativa de las características generales de los diferentes sistemas biométricos (2)

2.1.7. Usos y aplicaciones de la biometría

La necesidad de seguridad se ha disparado con el auge de Internet, las compras o transacciones bancarias on-line, o los atentados. La biometría se erige como el futuro de los sistemas de seguridad y su desarrollo en los últimos años ha experimentado un crecimiento geométrico respecto a otras tecnologías de seguridad, debido principalmente por el avance tecnológico continuo en el que vivimos [4] [7].

En resumen, podemos dividir las aplicaciones de la biometría en tres grandes grupos:

- **Comerciales.** En este grupo tenemos por ejemplo los **ATMs** (Asynchronous Transfer Mode, modo de transferencia asíncrona), la **gestión de historiales médicos** y el llamado **Control de acceso físico o lógico** (empresas y administraciones públicas, edificios y hogares en general, ordenadores, teléfonos, PDAs, etc.). En cuanto a las **entidades financieras**, este es quizá uno de los sectores más preocupados históricamente por la seguridad, para evitar fraudes y pérdidas de dinero. Por ello, algunas entidades ya apuestan fuertemente por los sistemas biométricos:
“El Bank of America y en instituciones financieras como VISA o MasterCard ya se han implementado sistemas de reconocimiento manual y del iris”.

También podemos mencionar el **comercio electrónico y banca electrónica**. Ésta ha sido una de las áreas que más ha crecido en los últimos años, y la que más ha influido en el desarrollo de nuevos sistemas de seguridad, hasta el punto de que la idea en este sector es reducir los precios de venta de los dispositivos de reconocimiento biométrico hasta que vayan acabando integrados directamente como parte de todos los PC.

- **Gubernamentales.** El grupo presente incluye el carnet de conducir, las tarjetas de seguridad social, el control de pasaportes, la tarjeta de identificación nacional, etc. Con respecto a esto último, hablemos del **DNI electrónico**, un DNI biométrico que suponga la eliminación de tarjetas, sustituidas por ejemplo por el iris de su titular, sería un punto importante para la seguridad con respecto a la identificación personal en su país.

“La Cámara de Diputados y el Senado en Colombia para evitar fraude en las votaciones, recurrieron a identificar a sus ciudadanos mediante geometría de la mano”.

- Por parte del área de **turismo y viajes**, el reciente endurecimiento de varios gobiernos sobre la normativa para acceder a la zona de libre tránsito de los aeropuertos han generado la necesidad de buscar otros métodos de seguridad diferentes a los actuales.

“El aeropuerto británico de Heathrow, fue el escogido para realizar la implantación de un sistema de reconocimiento del iris y permite la identificación de los usuarios en tiempo real a través de un banco de datos previamente recogido”.

- **Forenses:** investigaciones policiales, identificación de terroristas identificación de personas desaparecidas...

Pero el sistema biométrico más extendido en la actualidad es la huella dactilar, que incluso ya algunos ordenadores portátiles incorporan lector o sensor de huellas para identificar al propietario debido a su bajo coste. Algunos lugares que controlan o han controlado la seguridad con este sistema biométrico son los siguientes:

- **Control de acceso a diferentes áreas en el Pentágono**
- **Acceso a computadoras de redes financieras en Italia**
- **Automated Banking Terminal en Australia**
- **Aduana e inmigración en Ámsterdam**
- **Control de acceso a la Expo'92 Sevilla**
- **Smartphones tales como los de Apple o Samsung**
- **Acceso a distintas aplicaciones móviles: Google Play Store, Telegram, PayPal, Journey...**

Tras analizar los diferentes sistemas biométricos que existen, viendo sus características y utilidades, con la gráfica que se muestra podemos comprobar que el sistema más utilizado en la actualidad por su sencillez, seguridad y coste es la huella dactilar [9]:

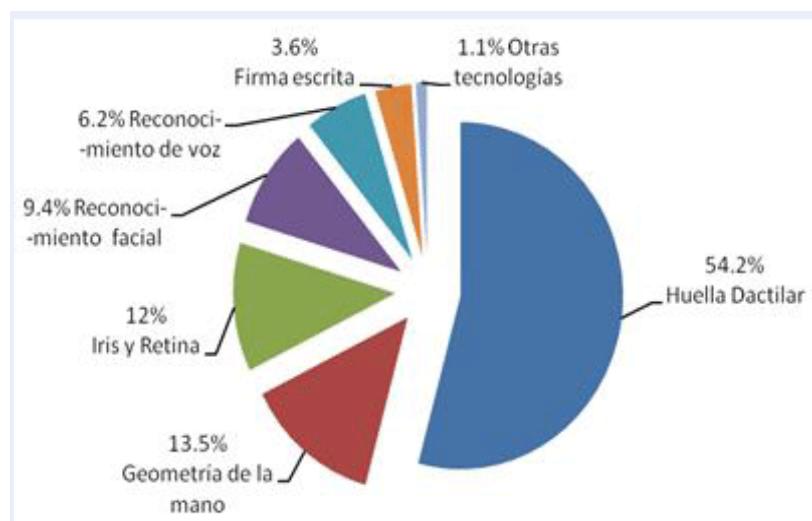


Figura 16: porcentaje de tecnologías biométricas más utilizadas. Análisis realizado por el International Biometric Group

2.2. Huella dactilar

2.2.1. Fundamentos de las huellas dactilares

El reconocimiento de huella dactilar es el método de identificación biométrica más sencillo, accesible y con importante aceptación por parte de los usuarios de todos los tipos de sistemas biométricos existentes.

La ciencia que se encarga de estudiar esta característica física recibe el nombre de **Dactiloscopia**, concepto que inauguró el doctor **Francisco Latzina**. Todos los sistemas dactiloscópicos tienen tres principios fundamentales: la perennidad, la inmutabilidad y la diversidad infinita [4] [9] [12].

Las huellas son características exclusivas de los primates. En los humanos, estas formas que adopta la piel en las yemas de los dedos no varían en sus características a lo largo de toda la vida, desde que se manifiestan a partir del sexto mes del desarrollo del embrión, y por ello se da la **perennidad**.

En cuanto a la **inmutabilidad**, el desarrollo físico y las enfermedades no afectan a las características de una huella dactilar. En caso de un desgaste involuntario, como puede ser una herida, el tejido epidérmico que la conforma se regenera tomando su forma original.

Finalmente tenemos el principio de la **diversidad infinita**. Con las características que contienen, cada huella constituye un patrón único para cada persona con la combinación de los rasgos que veremos posteriormente, hasta para los gemelos. Se estima así que la probabilidad de que dos individuos tengan las mismas huellas dactilares es sobre 1 en 64.000 millones.

Además, las huellas están constituidas por rugosidades que forman **salientes (crestas papilares)** y **depresiones (surcos interpapilares)**.

Las **papilas** son las pequeñas protuberancias que nacen en la dermis y sobresalen en la epidermis. Sus formas son muy variadas.

Las **crestas** son los bordes sobresalientes de la piel formados por una sucesión de papilas. En ellas se encuentran las glándulas sudoríparas. Este sudor contiene aceite por los surcos de la huella, y cuando el dedo contacta con una superficie, queda un residuo produciendo un negativo de la huella. Las figuras que conforman las crestas se llaman **dactilogramas** (dactilogramas papilares en nuestro caso). Se clasifican en:

- **Dactilograma natural.** Se encuentra en la yema del dedo, formado por las crestas papilares de manera natural.
- **Dactilograma artificial.** Dibujo provocado por entintar un dactilograma natural e imprimirllo en una zona idónea.
- **Dactilograma latente.** Es la huella que se deja por cualquier dactilograma natural al tocar un objeto o superficie. Queda marcado pero es invisible, por lo que se necesita de un reactivo para su observación.

Por otra parte, estos dactilogramas se dividen en tres partes:

- **Sistema dactilar marginal.** Son las crestas que forman el marco del dibujo dactilar. La cresta más interna se conoce como límite marginal, la cual conforma el delta por su cara exterior.
- **Sistema dactilar nuclear.** Son las crestas que conforman el núcleo o centro del dibujo dactilar. La cresta más externa se denomina límite nuclear, y forma el delta por su cara interna.
- **Sistema dactilar basilar.** Son las crestas que conforman la base del dibujo dactilar, llegan hasta el pliegue de flexión. La cresta más interna se denomina límite basilar, y forma el delta en su base.

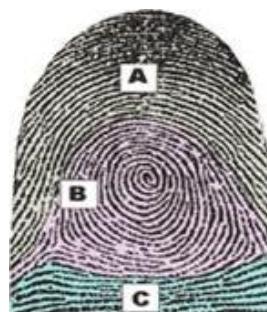


Figura 17: gráfico de una huella donde podemos ver el sistema dactilar marginal (A), nuclear (B) y basilar (C)

- **Surcos.** Son los espacios hundidos que se encuentran entre papila y papila. Producen los espacios en blanco que surgen al realizar la impresión de las huellas.
- **Poros.** Son los pequeños orificios situados en la cúspide de las crestas papilares o cerca de su vértice. Sirven para segregar el sudor y tienen formas muy distintas.

Existen cuatro grandes grupos de configuraciones dérmicas determinada como la **clasificación de Henry**. Para ello es necesario conocer los **Núcleos** y los **Deltas**, que son ciertas singularidades que se encuentran en determinadas huellas: áreas donde hay una triangulación o división de las líneas.

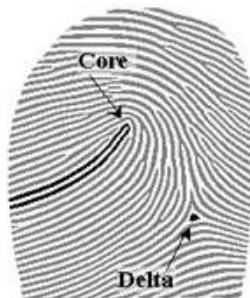


Figura 18: gráfico de una huella donde se indica un núcleo (core) y un delta

1. **Arco.** Es un dactilograma fundamental que carece de deltas y núcleos. Se caracteriza porque al principio las crestas son casi rectas, pero se van arqueando para dar lugar a un medio círculo.



Figura 19: huella dactilar con arcos

2. **Presillas internas.** Las crestas que forman su núcleo nacen en el costado izquierdo y recorren hacia la derecha, dando la vuelta después sobre sí mismas y regresando al punto de partida. Tienen un punto Delta al lado derecho de la imagen.



Figura 20: huella dactilar con presillas internas

3. **Presillas externas.** También disponen de un punto Delta, pero este es a la izquierda. Las crestas del núcleo nacen a la derecha y su recorrido es a la izquierda, dando la vuelta sobre sí mismas y regresando al punto de partida.



Figura 21: huella dactilar con presillas externas

4. **Verticilo.** Cuentan con dos puntos Delta, uno a la izquierda y otro a la derecha. Su núcleo puede adoptar formas circulares, elípticas y espirales. Pueden existir trideltos: verticilos con tres deltas.



Figura 22: huella dactilar con verticilos

En ciertos puntos, las líneas de la huella se corta de manera brusca o se bifurcan, produciendo las llamadas **minucias**. Existen diferentes puntos que caracterizan a una huella: empalme, convergencia, ojal, desviación, transversal, fragmento, abrupta, bifurcación, interrupción y punto.



Figura 23: ejemplos de minucias

Pasemos ahora a clasificar los diferentes tipos de huellas. En un nivel grueso, existen las huellas de tipo: **lazo derecho**, **lazo izquierdo**, **circular**, **arco** y **arco con tendencia**.

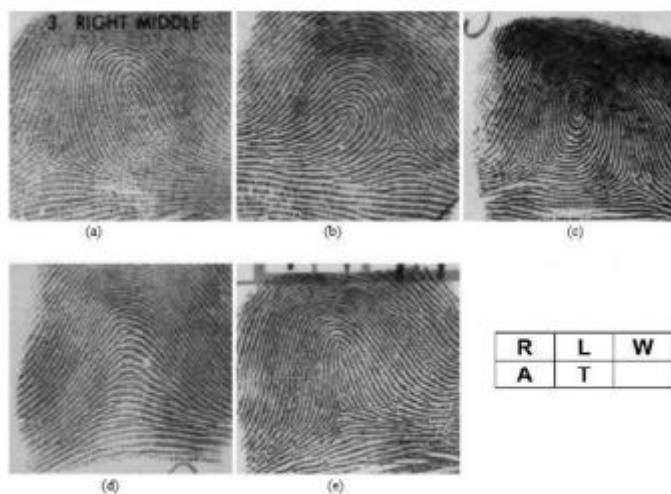


Figura 24: tipos de huellas: lazo derecho (a), lazo izquierdo (b), circular (c), arco (d) y arco con tendencia (e)

En un nivel más fino, se compara con el subconjunto de la base de datos que contiene únicamente ese tipo de huella dactilar. Esto requiere algoritmos para identificar a cuál de estos tipos pertenece una huella especificada.

A la hora de clasificar las huellas dactilares de manera automática, es necesario extraer las minucias de las imágenes de entrada, y para ello se necesita una cierta calidad de imagen.

Para contrarrestar esto y asegurar la identificación o verificación de la huella, es imprescindible realizar un algoritmo del realce de ésta en el módulo de la extracción de las minucias. Así, se mejora la claridad de las estructuras de la cresta y del surco de las imágenes de las huellas dactilares de entrada.

2.2.2. Procesamiento de la huella digital

Los pasos para el procesamiento de la huella dactilar mediante un sistema automatizado son:

- 1. Mejora de la imagen.** Básicamente se trata de eliminar las zonas confusas de la imagen original (ruido).
- 2. Binarización.** Se pasa la imagen original en tonos de gris a blanco y negro.
- 3. Adelgazamiento.** Las crestas líneas dactilares se procesan para que tengan el mismo grosor, y así se facilita la identificación de los puntos característicos.
- 4. Extracción de puntos característicos.** Se detectan y se extraen las posiciones exactas de cada punto característico. Al digitalizar una huella, los detalles de cada línea y, teniendo en cuenta las posiciones de las minucias existentes, unos algoritmos procesan toda esta información y otorgan un índice numérico a esta

huella. Cuando un usuario intenta identificarse, coloca su dedo sobre un lector de huella digital y ésta es escaneada y analizada para comprobar que existe en la base de datos su homóloga. Existen dos grandes categorías en cuanto a técnicas con las que se comparan huellas dactilares:

- a. **Técnica de puntos Minutia.** Este método no considera el patrón global de crestas y surcos.
 - i. Encuentran las minucias si la huella dactilar no es de baja calidad.
 - ii. Procede a su colocación relativa en el dedo.
- b. **Técnica de correlación.** La técnica de correlación requiere de una localización precisa de un punto de registro y se ve afectada por el desplazamiento y rotación de la imagen.

Además, en esta etapa se realiza la identificación y verificación una vez que se tiene dicho índice numérico. Se llevan a cabo búsquedas 1:1 para verificar la identidad de un individuo, o 1:N para identificarla.

Cabe destacar que la cantidad mínima de puntos característicos necesarios para proceder a comparaciones eficaces es de quince. Esta extracción de puntos se puede llevar a cabo con diversas técnicas, como pueden ser: la extracción de puntos característicos desde la imagen de la huella, mediante un banco de filtros de Gabor o sobre la propia imagen de la huella en escala de grises.

2.3. Tipos y funcionamiento de los distintos lectores de huella dactilar

El lector de huella digital, en primer lugar, obtiene una imagen de la huella y después compara el patrón obtenido a partir de dicha imagen con los patrones de las huellas ya almacenadas (1:N).

Una vez que se ha leído la huella digital, los lectores necesitan analizar la imagen obtenida. Para reducir el consumo del procesador, se comparan rasgos específicos de la imagen, conocidos como **minutiae** o **minucias**, y no se analiza la imagen al completo.

El software del lector utiliza algoritmos muy complejos para reconocer y analizar las minutiae. Se miden las posiciones relativas de éstas, y si dos imágenes tienen terminaciones de crestas y bifurcaciones que forman la misma figura dentro de la misma dimensión, existe una gran probabilidad de que sean del mismo individuo. Para obtener una coincidencia, el sistema del lector no necesita encontrar el patrón al completo de minutiae en la muestra y en la imagen almacenada, simplemente debe encontrar un número suficiente de patrones que tengan en común, variando esta cantidad según el programador [13] [14] [15].

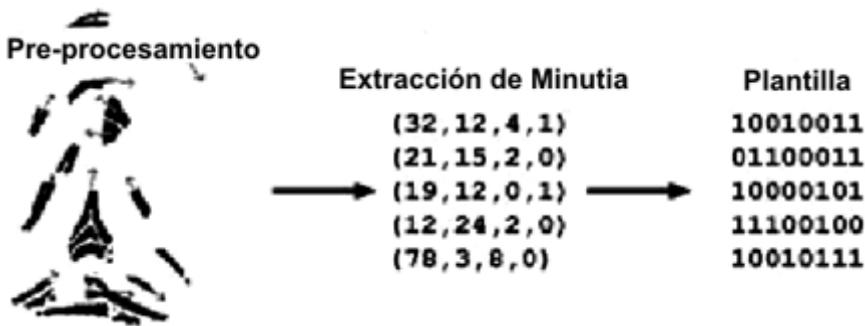


Figura 25: resumen de los pasos necesarios para el procesamiento de una huella

Existen varios métodos con los que se puede obtener esta imagen [6] [10]:

2.3.1. Lectores ópticos

Estos lectores funcionan con un dispositivo **CCD** (*Charged Coupled Device*), similar al que utilizan las cámaras digitales. Contienen unos diodos sensibles a la luz que generan una señal eléctrica. Cada uno de ellos graba un píxel que representa la luz que le es reflejada. Así, con el conjunto de luces y oscuridades se conforma una imagen de la huella leída.

El lector tiene su propia fuente de iluminación, como pueden ser unos LEDs, que iluminan las crestas de la huella digital apoyadas sobre la superficie de toque. Cuando las huellas del dedo tocan esta superficie, la luz se absorbe y entre los distintos niveles de la huella se produce la reflexión. El CCD genera una imagen invertida del dedo, en la que las zonas oscuras (más luz reflejada) definen las crestas, y las más claras representan los surcos entre las crestas.

Para impedir que la imagen obtenida sea de baja calidad o resolución, el CDD comprueba la oscuridad promedia de los píxeles (mientras más sensores o **PPI** (*point per inch*) tenga el escáner, más seguro será), o los valores generales en una pequeña muestra. De esta forma, se rechazan las imágenes que son muy oscuras o muy claras, dado que pueden inducir a errores.

Con la imagen correcta, el lector revisa la definición de la imagen, y el procesador ahora intentará solucionar el problema de la orientación del dedo incorrecta, ya que las coordenadas de las minucias van a ser distintas.

Para corregir la orientación, hay que ver en qué sentido están orientadas las líneas. Para ello se tiran líneas horizontales y verticales desde el centro de la huella, y se va mirando el ángulo que forman las líneas del dedo con esas rectas. Obtenemos una serie de ángulos invariables y que siempre están en el mismo orden. De esta forma, la próxima vez que se lea una huella, se mide de nuevo los ángulos, que serán los mismos y en igual orden a la huella buscada.

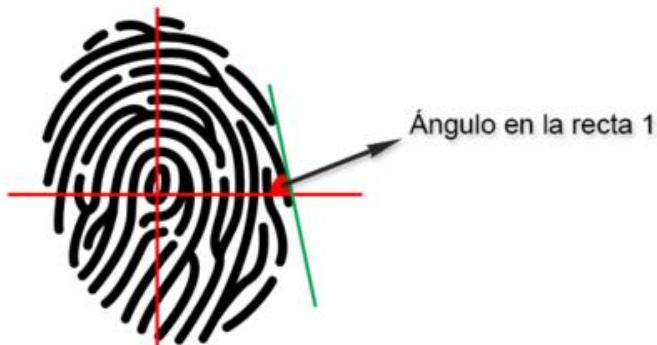


Figura 26: corrección de la orientación de la huella para lectores ópticos

El principal inconveniente de estos escáneres es que la calidad y rugosidad de la piel afecta al resultado. Es decir, influyen parámetros como la suciedad y el polvo, la humedad o sequedad del dedo, piel arrugada, etc.

Sin embargo, estos lectores ópticos que se han descrito serían los llamados lectores de huellas dactilares ópticos reflexivos, pero existen los lectores ópticos transmisivos con los que no se requiere el contacto directo del dedo con la superficie que cubre el sensor. En estos escáneres, que ofrecen una ventaja con respecto al siguiente tipo, una luz atraviesa el dedo desde un lado, mientras que al otro lado una cámara enfoca directamente la huella y captura la imagen de esta [18].

2.3.2. Lectores de capacitancia

Generan la imagen de la huella digital mediante corriente eléctrica en lugar de realizarlo con luz, como lo hacen los escáneres ópticos. Esto hace que el sistema sea más difícil de engañar ya que no son impresiones visuales de luz de las huellas digitales, como los lectores ópticos reflexivos, que se podrían vulnerar con imágenes.

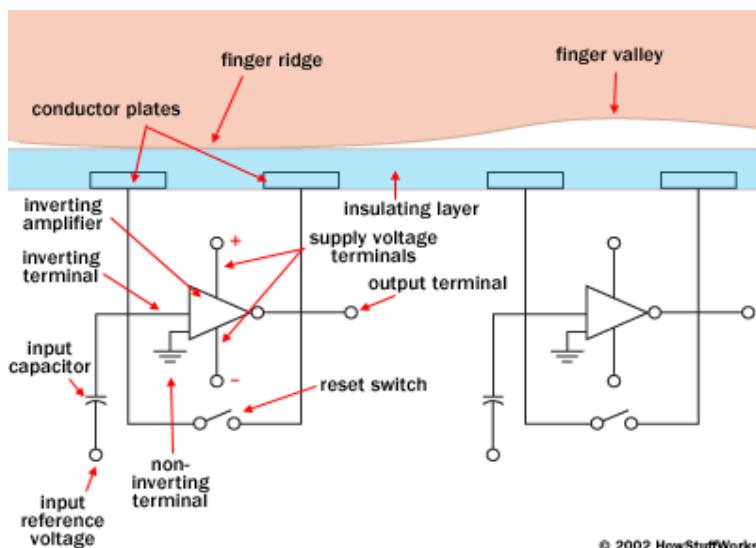


Figura 27: proceso de reconocimiento de una huella en lectores de capacitancia

Los sensores capacitivos se componen de chips que contienen numerosas celdas diminutas, más pequeñas que el ancho de una cresta, y cada una de ellas incluye dos placas conductoras cubiertas con una capa aislante. Estos escáneres aprovechan que la piel es eléctricamente conductora, y cuando acercamos el dedo, el campo eléctrico cambia.

El sensor se conecta a un integrado: circuito eléctrico construido sobre la base de un amplificador operacional inversor que altera un flujo de corriente. La alteración se basa en el voltaje relativo de dos fuentes, llamado terminal inversor (conectado a una fuente de voltaje de referencia y un bucle de retroalimentación que incluye las dos placas conductoras) y no-inversor (conectado a tierra). El terminal inversor funciona como un condensador, es decir, puede almacenar una carga, y la superficie del dedo actúa como una tercera placa condensadora, separada por las aislantes en la estructura de la celda. En el caso de los surcos de la huella, estos se interpretan como una bolsa de aire.

Al variar la distancia entre los condensadores moviendo el dedo más cerca o más lejos de las placas conductoras (de esta forma la salida de voltaje de un surco será diferente a la de una cresta), varía la capacidad eléctrica total del condensador. Así, el condensador en una celda bajo una cresta tendrá una capacidad eléctrica mayor que el condensador en una celda bajo un surco.

El procesador del lector lee la salida de voltaje y determina si es característico de una cresta o un surco. Al leer cada celda en los sensores, el procesador construye una imagen de la huella.

2.3.3. Lectores mecánicos

Están compuestos por miles de transductores de presión muy pequeños, puestos sobre una superficie y que reaccionan a la presión del dedo.

Un diseño alternativo utiliza commutadores que están cerrados cuando son presionados por una cresta, pero permanecen abiertos cuando están bajo un surco. Esto sólo proporciona un bit de información por píxel, en lugar de trabajar con una escala de grises.

2.3.4. Lectores térmicos

Este tipo de sensor es capaz de detectar el calor del dedo y capta la forma de la huella dactilar, gracias a que las crestas conducen más calor que los surcos del dedo. Un cambio de temperatura se traduce en un cambio en la distribución de carga de su superficie. La imagen está en la escala de grises que tiene la calidad adecuada incluso con el dedo desgastado, con suciedad, con grasa o con humedad.

2.3.5. Lectores LE (*Light Emitting*)

Es una nueva tecnología que supera de forma notable las tecnologías de lectura de huella dactilar anteriores. Está basado en un polímero que reacciona al contacto con

la huella emitiendo luminiscencia. De esta forma, al ser tan sólo la parte que contacta con el sensor la que emite luz, las crestas de la huella dactilar forman un patrón. El sensor LE genera una imagen exacta de la huella, indistintamente que dicha huella esté sucia, o incluso pintada.

Incorpora además la tecnología **LFD** (*Life Finger Detection*), que distingue entre una huella dactilar real, respecto a la imagen de una huella dactilar.

2.3.6. Lectores de salida dinámica

La mayoría de los sensores descritos han sido ya alterados de alguna forma. Para evitar esto, se ha añadido un nuevo modo de funcionamiento. En lugar de colocar sencillamente el dedo de forma estática sobre el sensor, el dedo se desplaza lentamente a lo largo del mismo. El sensor sólo dispone de una estrecha zona sensible, y genera una secuencia completa de imágenes, las cuales pueden ser re-ensambladas, mediante un procesador, en una imagen completa.

2.4. Ejemplos de sistemas de huella dactilar

Una vez que hemos explicado el funcionamiento de los diferentes tipos de lectores de huella dactilar, pasamos a analizar algunos de los escáneres más importantes que existen en el mercado [19] [20].

2.4.1. DigitalPersona U.are.U

La tecnología de DigitalPersona está reconocida actualmente como una de las mejores del mundo, e incluye todos los componentes necesarios de hardware y software para poder ofrecer una solución completa para cualquier tipo de negocio, incluyendo soluciones de identificación vía web. Hay varios casos de éxitos probados en México y más de 70.000 sensores biométricos instalados a la fecha. Esto demuestra la calidad de DigitalPersona.

Existen dos versiones principales: DigitalPersona U.are.U 4000B, y el nuevo DigitalPersona U.are.U 4500. Características [21]:

DigitalPersona 4500	
Tipo	Óptico
Sistemas operativos	Windows, Windows Server, Linux
Imagen de captura	Escala de gris de 8 bits. Encriptación de los datos de las huellas
Área de captura y peso	14.6 x 18.1 mm 105 gramos
Resolución (píxeles)	512 dpi
Interfaz	USB 2.0 Full-speed High power device
Voltaje	5.0 V ± 5% suministrado por el USB

Otras características	LED azul Formato compacto Excelente calidad de imagen Alta resistencia ESD
------------------------------	---

Tabla 3: características DigitalPersona 4500



Figura 28: Digital U.are.U Persona 4500

2.4.2. Lumidigm

Los sensores de huellas dactilares de la serie V de Lumidigm superan a otros sensores al mejorar el rendimiento, la precisión, la velocidad y la seguridad de su aplicación.

Utiliza la tecnología de imágenes multiespectrales, y lee de manera simultánea la superficie y la subsuperficie para captar imágenes nítidas y limpias en todo momento, aunque las características superficiales estén ausentes o sean difíciles de distinguir debido a factores tales como condiciones ambientales o de la piel, edad, suciedad y presión del dedo.

Las tecnologías actuales suelen tener un grado de error hasta de un 16%, pero Lumidigm dice garantizar la reducción de huellas digitales inadecuadas a menos de 1%.

Los principales lectores que siguen esta tecnología son el Sensor Mercury y el Sensor Venus. Vamos a describir las características de este último [22]:

Sensor Venus	
Tipo	Óptico
Sistemas operativos	Windows (32 y 64 bits) y Linux (32 bits)
Imagen de captura	Escala de gris de 8 bits
Área de captura y peso	17 X 28 mm (0,7" x 1,1", elipse)
Resolución (píxeles)	500 dpi
Interfaz	USB 2.0 ó RS-232
Voltaje	+5 VDC 500mA (máximo)
Otras características	Resistencia ESD de 15 kV Iluminación multiespectral

Tabla 4: características Sensor Venus



Figura 29: Lumidigm Venus Series sensor

2.4.3. Nitgen Fingkey Hamster

Los lectores de huella digital Fingkey Hamster son unos sensores de última generación cuya función principal es evitar huellas falsas, y puede conectarse a un ordenador para suplantar cualquier área que requiera de una contraseña por identificación de huella dactilar. Su motor de comparación de huellas está catalogado como el número uno según la FVC (*Fingerprint Verification Competition*). Vamos a describir el lector de huellas digital NITGEN Fingkey Hamster II DX [23]:

Fingkey Hamster II DX	
Tipo	Óptico
Sistemas operativos	Windows (32 y 64 bits) y Linux (kernel superior a 2.6)
Imagen de captura	248 x 292 píxeles
Área de captura y peso	17 x 20 mm 100 gramos
Resolución (píxeles)	500 dpi
Interfaz	USB 2.0 High/Full Speed, dispositivo plug and play
Voltaje	+5 V
Otras características	Pequeño, ágil y versátil Función de encendido automático Sensor resistente a rayones, impactos, vibraciones y descargas electroestáticas

Tabla 5: características Fingkey Hamster II DX



Figura 30: NITGEN Fingkey Hamster

2.4.4. ATMEL Fingerchip

La familia de sensores de ATMEL Fingerchip son unos escáneres térmicos que utilizan el método de captura “*swipe*”, es decir, arrastrar el dedo a través del sensor. Son escáneres de muy pequeño tamaño pero de precios económicos. Los rasgos generales de estos lectores son los siguientes [24]:

ATMEL Fingerchip	
Tipo	Térmico
Sistemas operativos	Windows (32 bits)
Imagen de captura	232 x 8 píxeles
Área de captura y peso	11.6 x 0.4 mm
Resolución (píxeles)	500 dpi
Interfaz	Serial (mayor de 16 Mbps)
Voltaje	+5 V
Otras características	Tamaño diminuto Captura “swipe”

Tabla 6: características ATMEL Fingerchip



Figura 31: ATMEL Fingerchip

2.4.5. Fulcrum Biometrics

El proveedor de Fulcrum Biometrics es un líder internacional en distribución e integración de varios sistemas biométricos.

En nuestro ámbito, Fulcrum Biometrics ofrece dispositivos portables biométricos como el FbF MobileOne para realizar lectura de huellas digitales. Está construido exclusivamente para la plataforma iOS (iPod Touch, iPhone), y ha sido certificado por el FBI. Especificaciones [25]:

Fulcrum Biometrics FbF MobileOne	
Tipo	Óptico
Sistemas operativos	Apple iOS (generaciones de iPod a partir de la segunda en adelante, iPhone desde la versión 3G)
Imagen de captura	360 x 256 píxeles, escala de gris de 8 bits

Área de captura y peso	18 x 12.8 mm
Resolución (píxeles)	508 dpi
Interfaz	Wi-Fi 802.11 b.g.n Minipuerto USB Bluetooth Conector Apple de 30 pines
Voltaje	+5 V a través del minipuerto USB
Otras características	Menos de 2 segundos para autenticarse Incluye una batería interna recargable Conexión local o remota

Tabla 7: características Fulcrum Biometrics FbF MobileOne



Figura 32: Fulcrum Biometrics FbF MobileOne

2.4.6. UPEK TouchChip

UPEK TouchChip desarrolla los principales lectores de huella digital a través de capacitancia. Se suelen utilizar en notebooks, teclados, cerraduras de puertas, terminales de puntos de venta (TPV), PDAs... Características del UPEK TouchChip TCRU1C [26]:

UPEK TouchChip TCRU1C	
Tipo	Capacitancia
Sistemas operativos	Windows, Linux y Mac OS X (32 y 64 bits) y Android (acceso root)
Imagen de captura	256 x 360 píxeles
Área de captura y peso	13 x 18 mm 40 gramos
Resolución (píxeles)	508 dpi
Interfaz	USB 2.0
Voltaje	+5 V
Otras características	Cuenta con una superficie de recubrimiento relativamente dura

Tabla 8: características UPEK TouchChip TCRU1C



Figura 33: UPEK TouchChip TCRU1C

2.4.7. Lector de huellas GT-511C3

Este tipo de lector de huellas, que permite almacenar hasta 200 huellas, ofrece la posibilidad de conectarse a un microcontrolador, como puede ser una placa Arduino (compañía de hardware libre), o directamente a nuestro ordenador mediante un conversor serie-USB FTDI, con lo que podremos hacer uso del software que proporciona.

Cuenta con una pequeña CPU de 32 bits ARM Cortex M3 Core con la que procesa los datos. Es un escáner sencillo y muy económico, cuyas algunas especificaciones son [27] [28]:

Fingerprint Scanner GT-511C3	
Tipo	Óptico
Sistemas operativos	Open Source
Imagen de captura	202 x 258 píxeles
Área de captura y peso	14 x 12.5 mm
Resolución (píxeles)	450 dpi
Interfaz	Protocolo Serie UART (9600 baud por defecto) USB 1.1 Full Speed
Voltaje	5-6 V
Otras características	Puede almacenar hasta 200 huellas diferentes y realizar Alta velocidad y precisión gracias al algoritmo SmackFinger 3.0 Puede descargar las imágenes desde el módulo

Tabla 9: características Fingerprint Scanner GT-511C3



Figura 34: Fingerprint Scanner GT-511C3

2.4.8. Sensor de huellas ADAFRUIT con Arduino

La compañía de Industrias Adafruit, especializada en desarrollar hardware *open-source*, tiene en su lista de productos un sensor de huellas el cual podemos hacerlo funcional mediante una placa Arduino o cualquier otra placa microcontroladora o tarjeta de desarrollo de código abierto, ya que el dispositivo funciona con el protocolo serial.

Es la opción más económica de todas con la que podemos hacer pleno uso de las ventajas que ofrece la identificación por huella dactilar. El sistema realiza procesamiento digital de imágenes interno con un DSP (*Digital Signal Procesor*), además de incluir capacidades de comparación en base de datos y actualización de la misma [29] [30].

Adafruit Fingerprint Sensor	
Tipo	Óptico
Sistemas operativos	Open-Source
Imagen de captura	256 x 288
Área de captura y peso	14 x 18 mm 20 gramos
Resolución (píxeles)	500 dpi
Interfaz	TTL Serial
Voltaje	3.6 - 6.0 VDC
Otras características	Puede almacenar hasta 162 huellas diferentes y realizar 5 niveles de seguridad Tiempo de adquisición menor de 1 segundo Baud Rate: 9600,19200,28800,38400,57600 (57600 por defecto)

Tabla 10: características Adafruit Fingerprint Sensor



Figura 35: Adafruit Fingerprint Sensor

Al margen de todos estos lectores que hemos visto, existen muchísimos tipos de sensores más que realizan la misma función de identificación por huella dactilar. Varios de ellos son también muy importantes, por lo que al menos vamos a mencionarlos ya que no han tenido cabida en esta comparativa: **Digid Mini** y **Digid XS** de **Identificacion International**, el **AFS510** de **ARH** (*Adaptive Recognition Hungary*), el proveedor **Crossmatch Technologies** con sus sensores **Verifier 300LC, 310LC** o **LScan Guardian**, los **Suprema BioMini** centrados en su utilidad en el sistema Android, y, por último, el lector de huellas **AuthenTec AF-S2** cuyo sensor es de campo eléctrico [31].

2.5. Tipos de SDKs para el desarrollo de aplicaciones basadas en sistemas de huella dactilar

Para hacer funcional a un lector de huellas digital, es necesario la utilización del algún **SDK** (*software development kit*) compatible, es decir, tener una interfaz de programación de aplicaciones con la que se pueda interactuar con la parte hardware del sistema. Básicamente consiste en un conjunto de herramientas de desarrollo de software con las que podemos crear aplicaciones y añadir funcionalidades al sistema digital de lector de huellas que hayamos elegido [32].

A continuación vamos a describir un conjunto de tecnologías software o SDK útiles para el reconocimiento de huella digital:

ZK Software: Biokey SDK [33]	
Plataformas	Windows, UNIX, Linux, Palm...
Lenguajes de programación	Visual C++ 6.0, C++ Builder 5.0, Viusal Basic 6.0, Delphi 5.0
Identificación y verificación	Identificación 1:N de hasta 10.000 por segundo, verificación 1:1
Compatibilidad con lectores	Media
Almacenamiento	Ilimitado
Otras características	Clasificación automática de huellas digitales

Tabla 11: características del Biokey SDK

DigitalPersona Pro Enterprise [34]	
Plataformas	(Microsoft) - XP/Vista/Win 7 (32/64), XP Embedded (32-bit), Windows Server 2003/2008 (32/64-bit), Citrix/Terminal Services (local driver)
Lenguajes de programación	.NET, Java, C#, ActiveX/COM, C/C++, JPOS/OPOS
Identificación y verificación	Identificación 1:N y autenticación secundaria opcional (<i>Single Sign-On</i>)
Compatibilidad con lectores	Alta
Almacenamiento	Posibilidad de usar cualquier base de datos que elija el desarrollador. Almacenamiento ilimitado
Otras características	Cifrado de disco completo

Tabla 12: características del SDK DigitalPersona Pro Enterprise

VeriFinger SDK [35]	
Plataformas	Microsoft Windows, Linux y Mac OS X (32 y 64 bits)
Lenguajes de programación	C/C++, C#, Visual Basic .NET, Sun Java 2
Identificación y verificación	Verificación 1:1 e identificación 1:N
Compatibilidad con lectores	Muy alta
Almacenamiento	Ilimitado
Otras características	Incluye en su versión extendida aplicaciones cliente de ejemplo, y un servidor de comparación listo para usar. Velocidad de comparación de 40.000 huellas digitales por segundo

Tabla 13: características del VeriFinger SDK

IDKit SDK [36]	
Plataformas	OS, Windows (32-64 bits), Linux (32-64 bits), Android, iOS 6
Lenguajes de programación	C, C++, VB, C#, Java
Identificación y verificación	Verificación 1:1 e identificación 1:N
Compatibilidad con lectores	Trabaja con todo tipo de lectores que obtienen imágenes tipo: BMP, RAW, JPEG, JPEG2K, GIF, PNG
Almacenamiento	Ilimitado
Otras características	Integración e implementación fácil y rápida

Tabla 14: características del IDKit SDK

Free Fingerprint Verification SDK [37]	
Plataformas	Microsoft Windows de 32 bits
Lenguajes de programación	C++, C#, Visual Basic .NET, Visual Basic 6, Sun Java 2, Delphi 7
Identificación y verificación	Sólo verificación dactilar 1:1
Compatibilidad con lectores	Muy alta
Almacenamiento	Hasta 10 registros, en una base de datos propia
Otras características	Gratis

Tabla 15: características del Free Fingerprint Verification SDK

BioMini SDK [38]	
Plataformas	OS, Windows (32-64 bits), Linux (32-64 bits), Android, iOS 6
Lenguajes de programación	C/C++ (Linux), Java (Android), Visual C++, Visual Basic, C#, VB.NET, Java (Windows)
Identificación y verificación	Verificación 1:1 e identificación 1:N. Rápida velocidad de verificación: 1:100.000 en un segundo
Compatibilidad con lectores	Limitado a los lectores propios de BioMini y SFU
Almacenamiento	Ilimitado
Otras características	Rápida velocidad de verificación: 1:100.000 en un segundo

Tabla 16: características del BioMini SDK

Lectores de huellas con plataforma Arduino [29]	
Plataformas	OS, Windows (32-64 bits), Linux (32-64 bits)
Lenguajes de programación	C/C++ (Arduino IDE)
Identificación y verificación	Verificación 1:1 e identificación 1:N
Compatibilidad con lectores	Limitado a los lectores compatibles con placas microcontroladoras Arduino
Almacenamiento	162 registros
Otras características	Código abierto. Tiempo de 1 segundo para obtener la imagen de una huella

Tabla 17: características del software necesario para los lectores de huellas con plataforma Arduino

Capítulo 3: Metodología de Desarrollo

Contenidos

Capítulo 3: Metodología de Desarrollo.....	40
3.1. Introducción	40
3.1.1. Etapas del modelo iterativo e incremental	41
3.1.2. Ventajas y desventajas del modelo iterativo e incremental.....	42

Este capítulo contiene la descripción de la metodología de desarrollo que se ha seguido para llevar a cabo este Trabajo Fin de Grado.

Durante el desarrollo del proyecto se empleará una **metodología iterativa o incremental** como marco de trabajo para estructurar, planear y controlar el proceso de desarrollo de los diferentes subsistemas que componen el sistema. En los siguientes apartados profundizaremos sobre esta metodología.

3.1. Introducción

El desarrollo iterativo o incremental es un proceso de desarrollo de software que consiste en la iteración de varios ciclos de vida en cascada. Lo que propone este modelo es ir entregando un proyecto por pequeños módulos. Para saber cuántos módulos debemos entregar, tenemos que realizar nuestro análisis de requisitos, y de esta manera conseguiremos que el cliente tenga múltiples entregas de su proyecto y nos brinde así retroalimentación, consiguiendo que cada entrega sea más completa que la anterior [39].

De esta forma, al final de cada iteración se entrega una versión funcional del producto mejorada, desarrollándose el sistema poco a poco [40].

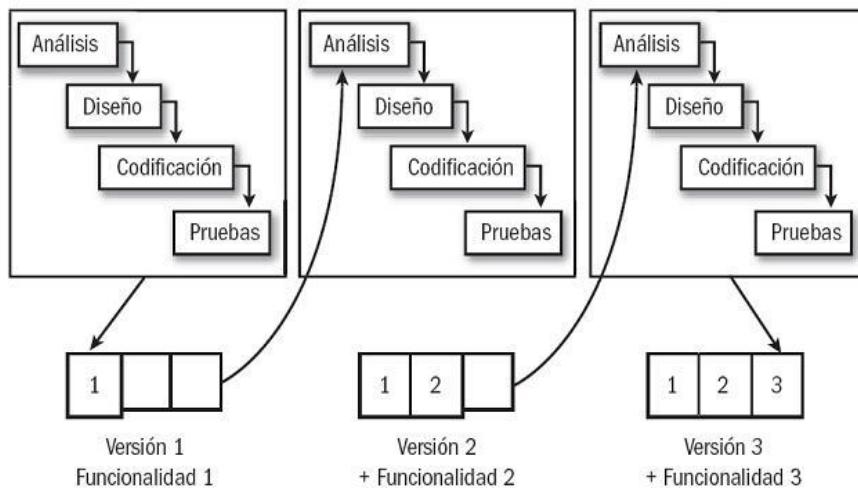


Figura 36: Modelo de ciclo de vida iterativo e incremental

Los pasos claves en el proceso son iniciar con una implementación simple de los requerimientos del sistema, e iterativamente mejorar la serie evolutiva de versiones hasta que el sistema completo este implementado. En cada iteración, se realizan cambios en el diseño y se agregan nuevas funcionalidades y capacidades al sistema. El proceso en sí mismo consiste en crear una versión del mismo. La meta de esta etapa es crear un producto con el que el usuario pueda interactuar, y por ende retroalimentar el proceso siguiente [41].

3.1.1. Etapas del modelo iterativo e incremental

Este modelo se puede dividir en los siguientes procesos [42]:

- **Etapa de inicialización.** El objetivo de esta etapa es la construcción de un producto en el cual se pueda obtener retroalimentación de parte del usuario final.
- **Etapa de iteración.** Consiste en el análisis, rediseño e implementación del producto de las iteraciones anteriores. Como se puede observar en la figura anterior, cada iteración se compone de distintas fases [43]:
 - **Análisis.** Lo primero que debemos hacer es averiguar qué es exactamente lo que tiene que hacer el sistema. La etapa de análisis en el ciclo de vida del software corresponde al proceso mediante el cual se intenta descubrir qué es lo que realmente se necesita y se llega a una comprensión adecuada de los requisitos del sistema.
 - **Diseño.** En la fase de diseño se han de estudiar posibles alternativas de implementación para el sistema que hemos de construir y se ha de decidir la estructura general que tendrá el sistema (su diseño arquitectónico).

- **Codificación.** Una vez que sabemos qué funciones debe desempeñar nuestro sistema (análisis) y hemos decidido cómo vamos a organizar sus distintos componentes (diseño), es el momento de pasar a la etapa de implementación. Antes de escribir código es fundamental haber comprendido bien el problema que se pretende resolver y haber aplicado principios básicos de diseño que nos permitan construir un sistema de información de calidad.
- **Pruebas.** La etapa de pruebas tiene como objetivo detectar los errores que se hayan podido cometer en las etapas anteriores del proyecto (y eventualmente, corregirlos). Lo suyo, además, es hacerlo antes de que el usuario final del sistema los tenga que sufrir. De hecho, una prueba es un éxito cuando se detecta un error.
- **Lista de control del proyecto.** Son las tareas que se crean y que describen las partes que conforman el proyecto. Se implementan y se rediseñan en cada etapa de iteración del producto.



Figura 37: etapas del modelo iterativo e incremental

3.1.2. Ventajas y desventajas del modelo iterativo e incremental

Las principales ventajas que ofrece utilizar este modelo iterativo e incremental son las siguientes:

- Resolución de problemas en tiempos tempranos.
- Visión de avance en el desarrollo.
- Aprendizaje y experiencia tras cada iteración.
- Provee de soporte para determinar la efectividad de los procesos y de la calidad del producto.
- Puede monitorearse el cambio relativo de varios aspectos de un producto o pueden proveer los límites de las medidas para apuntar a problemas potenciales y anomalías.

- Obtención del “*feedback*” o retroalimentación del usuario sobre un prototipo operativo.
- Permite manejar el riesgo del proyecto, apuntando a la resolución de los problemas por partes.

Sin embargo, este modelo puede presentar algunos inconvenientes, como pueden ser:

- Requiere de un cliente involucrado durante todo el curso del proyecto.
- Infunde responsabilidad en el equipo de desarrollo al trabajar directamente con el cliente.
- Sufre fuertes penalizaciones en proyectos en los cuales los requerimientos están previamente definidos.
- No garantiza por sí solo el éxito.
- Iteraciones costosas.
- Congelamiento de requerimientos.

Capítulo 4: Análisis del Sistema

Contenidos

Capítulo 4: Análisis del Sistema.....	44
4.1. Planificación del sistema.....	44
4.1.1. Viabilidad Técnica	44
4.1.2. Viabilidad Operacional	45
4.1.3. Viabilidad Económica	45
4.1.4. Viabilidad de la Solución	46
4.2. Análisis funcional.....	46
4.3. Diagramas de casos de usos detallados	47
4.3.1. Casos de uso detallados del actor Profesor	48
4.3.2. Casos de uso detallados del actor Administrador.....	51
4.4. Diagramas de actividades	53
4.5. Cronograma.....	54

Pasamos ahora a detallar el análisis del sistema a desarrollar en este cuarto capítulo, contemplando las siguientes secciones: la planificación del sistema, el análisis funcional, los detalles de cada caso de uso y los diagramas de secuencia.

4.1. Planificación del sistema

Tras haber analizado el objetivo de este Trabajo Fin de Grado, tratando de solventar los problemas surgidos en cuanto al control de asistencia del profesorado, vamos a analizar la viabilidad del sistema y las herramientas que se van a desarrollar.

Se creará una aplicación con la cual se registrará la asistencia de los profesores a clase y a tutorías mediante un sistema seguro y fiable, como es la lectura de su huella dactilar. Para ello se combinará con el sistema de reservas MRBS (explicado más adelante) y se podrá generar informes sobre las asistencias de cada profesor.

4.1.1. Viabilidad Técnica

Nuestro sistema contaría con un servidor (**Raspberry Pi**) donde se gestionen los fichajes en las distintas asignaturas de los profesores. Los fichajes se realizarán con un lector de huellas dactilares, y se interactuará con la parte hardware mediante un

entorno web con el que se podrá introducir y consultar toda la información sobre los fichajes.

Todo esto se demuestra que es viable, pues estas tecnologías no son innovadoras y llevan en uso bastantes años ya.

4.1.2. Viabilidad Operacional

Este proyecto dispondrá de una aplicación a través de la cual podremos gestionar los fichajes. Sólo se necesitará un navegador web para ejecutar la aplicación, y el dispositivo de fichaje solo requiere pasar un dedo de nuestra mano por el lector de huellas como método de identificación única. El sensor tiene que estar conectado a una placa microcontroladora (**Arduino**).

En nuestro caso, el servidor deberá estar en funcionamiento para que las aplicaciones web estén disponibles.

4.1.3. Viabilidad Económica

Brevemente vamos a comprobar la viabilidad económica del proyecto. Haremos uso de un lector de huellas que requiere su conexión con un microcontrolador (Arduino de **Grove**), siendo esto la manera más económica de disponer de un lector de huellas y SDK compatible con él.

En la siguiente tabla se indican los dispositivos necesarios para este trabajo fin de grado:

Dispositivo	Precio
Arduino Uno	21, 50 €
Grove Starter Kit for Arduino	44, 90 €
Grove Adafruit Fingerprint sensor	44, 90 €
Raspberry Pi 2 Model B	40, 00 €
TOTAL	151, 30 €

Tabla 18: cuantía total de los dispositivos necesarios

Y aquí se muestra el coste total del desarrollo de las aplicaciones:

Aplicación	Cargo	Horas	Salario / hora	Precio
Análisis del proyecto	Analista	60	20 € / hora	1200,00 €
Servidor	Administrador	5	12 € / hora	60,00 €
Aplicación Arduino	Programador	30	15 € / hora	450,00 €
Sistema de fichaje	Programador	50	15 € / hora	750,00 €
Servicio Web	Programador	160	15 € / hora	2400,00 €
Portal Web	Programador	40	15 € / hora	600,00 €
Generador de informes	Programador	20	15 € / hora	300,00 €
Pruebas	Calidad	10	13 € / hora	130,00 €
TOTAL		375		5.890,00 €

Tabla 19: coste total del desarrollo de las aplicaciones

Con esto, se puede observar que el coste del proyecto sería de unos 5890 € en total. Los dispositivos utilizados suman poco más de 150 €, algo que reduce muy significativamente la cantidad total si se utilizaría un lector de huellas más profesional con un SDK compatible.

4.1.4. Viabilidad de la Solución

Llegados a este punto, podemos observar que la solución propuesta es viable técnica, operacional y económica mente.

Los costes del trabajo son muy asequibles, excepto el desarrollo software del sistema. Sin embargo, una vez se hayan desarrollado, las aplicaciones no requerirán costes de funcionamiento, tan solo el mantenimiento de los dispositivos hardware.

4.2. Análisis funcional

Vamos a estudiar en este apartado las funcionalidades o requisitos que nuestro sistema requiere. Para ello haremos uso de los diagramas de caso de uso, con los cuales podemos representar la manera en la que los distintos actores (personajes o entidades que participan en un caso de uso) del sistema invocan funciones específicas (u otros pasos del procedimiento) para satisfacer los requerimientos del sistema [44].

Los módulos obtenidos para este proyecto son los siguientes:

- **Aplicación web con la que podamos registrar las huellas.** Esta aplicación recibirá los datos de la huella de un profesor que será registrada en el lector y los almacenará en una base de datos, así como los datos personales del usuario.
- **Aplicación web para fichar la entrada y salida a clase y a tutorías de un profesor,** con la cual el profesor a través del lector de huellas registrará la fecha y hora a la que asiste a una determinada clase o tutoría.

En cuanto a los actores o roles que existirán en esta aplicación serán:

- Un **usuario administrador** podrá generar informes acerca de la asistencia del profesorado a sus asignaturas y a sus tutorías, consultar todos los fichajes de los profesores, registrar nuevos usuarios, eliminar huellas registradas y configurar la conexión con el lector.
- **El profesor** tendrá acceso al sistema de reservas MRBS y al sistema de fichajes, por los cuales podrá guardar sus huellas, consultar sus fichajes, registrar sus tutorías, justificar una falta o solicitar alguna reserva.

En la siguiente imagen se reflejan los casos de uso globales para cada actor, en nuestro caso, los profesores y un administrador:

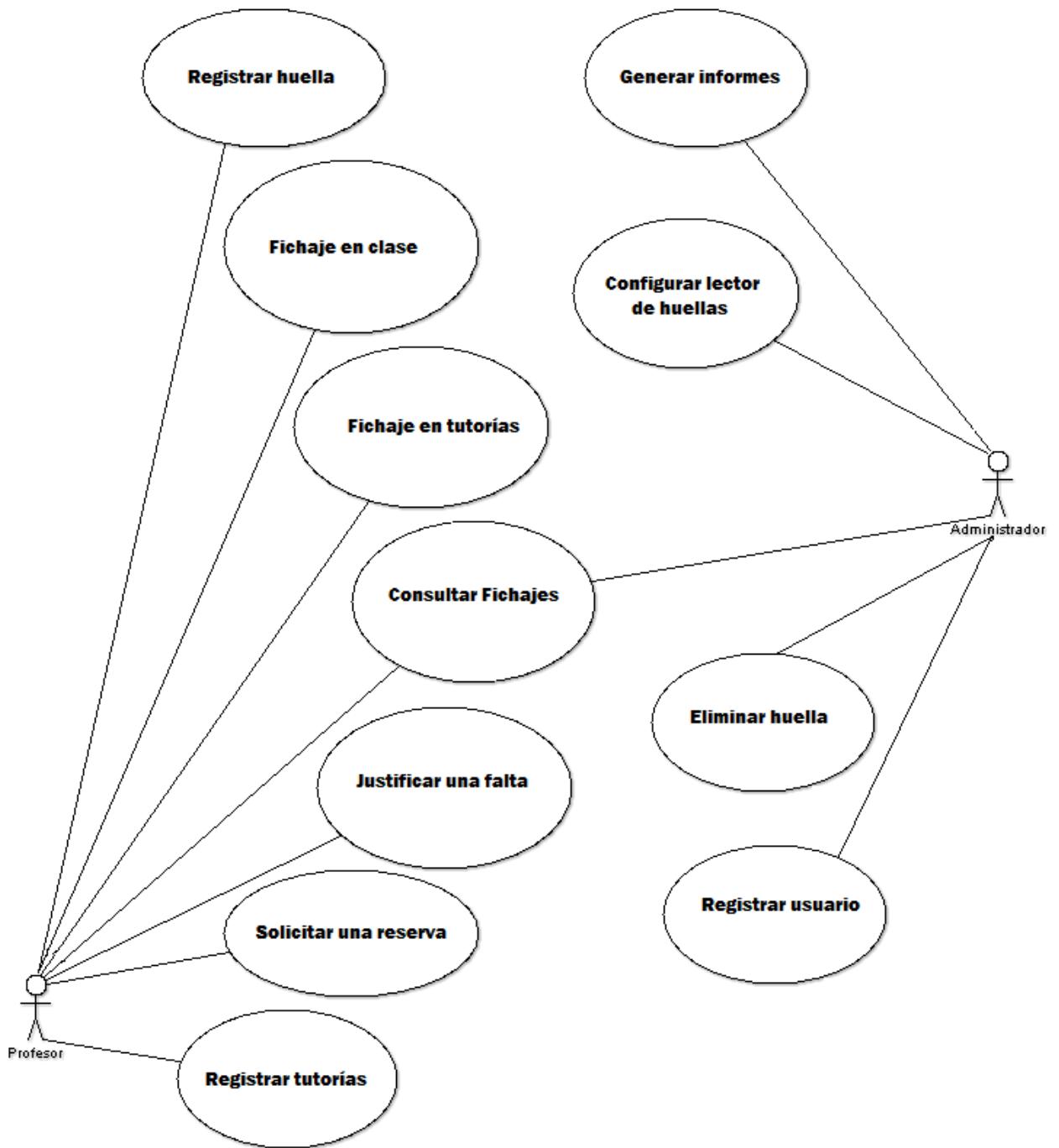


Figura 38: diagrama de casos de uso

4.3. Diagramas de casos de usos detallados

Una vez que conocemos el diagrama de casos de usos globales, pasamos a detallar cada caso de uso:

4.3.1. Casos de uso detallados del actor Profesor

NOMBRE:		Registrar huella
ACTORES	Profesor	
PRECONDICIONES	Tener el lector de huellas operativo Estar ya registrado en la plataforma	
POSTCONDICIONES		
FLUJO	NORMAL	ALTERNATIVO
	<ol style="list-style-type: none"> 1. Acceder a la opción “Registrar mi huella” de la web e identificarse 2. Apoyar un dedo en el sensor y esperar a que tome la imagen 3. Retirar el dedo del sensor 4. Apoyar nuevamente el mismo dedo en el sensor y esperar a que tome la imagen 5. Retirar el dedo 	<ol style="list-style-type: none"> 5. Si las dos imágenes de la huella no coinciden, habrá que repetir el proceso
DESCRIPCIÓN	El profesor guarda su huella en la base de datos para que pueda ser identificado posteriormente	

Tabla 20: caso de uso detallado "Registrar huella"

NOMBRE:		Fichaje en clase
ACTORES	Profesor	
PRECONDICIONES	Haber registrado su huella previamente Tener el lector de huellas operativo	
POSTCONDICIONES		
FLUJO	NORMAL	ALTERNATIVO
	<ol style="list-style-type: none"> 1. Llegar a la clase 15 minutos antes o después como máximo de la hora reservada, y apoyar un dedo en el sensor 2. Esperar a que el lector tome la imagen 3. Retirar el dedo del sensor cuando el lector haya encontrado alguna coincidencia 	<ol style="list-style-type: none"> 3. Si ocurre algún mensaje de error, retirar el dedo y probar de nuevo. 4. En caso de no haber encontrado alguna coincidencia, habrá que registrar nuestra huella
DESCRIPCIÓN	El profesor realiza el fichaje de asistencia a una clase dada	

Tabla 21: caso de uso detallado "Fichaje en clase"

NOMBRE:		Fichaje en tutorías
ACTORES	Profesor	
PRECONDICIONES	Haber registrado su huella previamente Tener el lector de huellas operativo	
POSTCONDICIONES		
FLUJO	NORMAL	ALTERNATIVO
	<ol style="list-style-type: none"> 1. Llegar a la tutoría correspondiente con un margen máximo de media hora y apoyar un dedo en el sensor 2. Esperar a que el lector tome la imagen 3. Retirar el dedo del sensor cuando el lector haya encontrado alguna coincidencia 	<ol style="list-style-type: none"> 3. Si ocurre algún mensaje de error, retirar el dedo y probar de nuevo. 4. En caso de no haber encontrado alguna coincidencia, habrá que registrar nuestra huella
DESCRIPCIÓN	El profesor realiza el fichaje de asistencia a una de sus tutorías	

Tabla 22: caso de uso detallado "Fichaje en tutorías"

NOMBRE:		Consultar fichajes
ACTORES	Profesor	
PRECONDICIONES	Estar registrado en la base de datos como un usuario Haber realizado algún fichaje previamente	
POSTCONDICIONES		
FLUJO	NORMAL	ALTERNATIVO
	<ol style="list-style-type: none"> 1. Entrar a la aplicación web accediendo mediante usuario y contraseña 2. Consultar los correspondientes fichajes en la opción "Fichajes" 3. Cerrar sesión 	<ol style="list-style-type: none"> 1. Mensaje de error en caso de no estar registrado o que el usuario y la contraseña no coincidan
DESCRIPCIÓN	El profesor puede consultar la lista de fichajes que ha realizado y comprobar que su asistencia ha sido registrada	

Tabla 23: caso de uso detallado "Consultar fichajes"

NOMBRE:	Justificar una falta	
ACTORES	Profesor	
PRECONDICIONES	Estar registrado y tener acceso al sistema	
POSTCONDICIONES		
FLUJO	NORMAL	ALTERNATIVO
	<ol style="list-style-type: none"> 1. Entrar a la aplicación web accediendo mediante usuario y contraseña 2. Acceder a la opción “Justificar falta” y seleccionar el fichaje que no ha sido justificado 3. Indicar en el formulario la “excusa” con la cual justifica la falta y el tipo de fichaje 	<ol style="list-style-type: none"> 1. Mensaje de error en caso de no estar registrado o que el usuario y la contraseña no coincidan
DESCRIPCIÓN	El profesor puede justificar una falta a clase o tutoría dando la excusa correspondiente	

Tabla 24: caso de uso detallado "Justificar una falta"

NOMBRE:	Registrar tutorías	
ACTORES	Profesor	
PRECONDICIONES	Estar registrado en la base de datos como un usuario	
POSTCONDICIONES		
FLUJO	NORMAL	ALTERNATIVO
	<ol style="list-style-type: none"> 1. Entrar a la aplicación web de MRBS accediendo mediante usuario y contraseña 2. Pulsar sobre la opción de registro o inserción de tutorías 3. Insertar la tutoría 4. Cerrar sesión 	<ol style="list-style-type: none"> 1. Mensaje de error en caso de no estar registrado o que el usuario y la contraseña no coincidan
DESCRIPCIÓN	El profesor registrará sus horarios de tutorías en la aplicación	

Tabla 25: caso de uso detallado "Registrar tutorías"

NOMBRE:	Solicitar reserva	
ACTORES	Profesor	
PRECONDICIONES	Estar registrado en la base de datos como un usuario	
POSTCONDICIONES		
FLUJO	NORMAL	ALTERNATIVO
	<ol style="list-style-type: none"> 1. Entrar a la aplicación web de MRBS accediendo mediante usuario y contraseña 2. Pulsar sobre la opción de solicitud de una reserva de espacio 3. Reservar el espacio 4. Cerrar sesión 	<ol style="list-style-type: none"> 1. Mensaje de error en caso de no estar registrado o que el usuario y la contraseña no coincidan
DESCRIPCIÓN	El profesor puede solicitar una reserva de un aula a través de la aplicación	

Tabla 26: caso de uso detallado "Solicitar reserva"

4.3.2. Casos de uso detallados del actor Administrador

NOMBRE:		Consultar fichajes
ACTORES	Administrador	
PRECONDICIONES	Estar registrado en la base de datos como un usuario administrador	
POSTCONDICIONES		
FLUJO	NORMAL	ALTERNATIVO
	<ol style="list-style-type: none"> 1. Entrar a la aplicación web accediendo mediante usuario y contraseña 2. Consultar todos los fichajes de los profesores en la opción “Fichajes” 3. Cerrar sesión 	<ol style="list-style-type: none"> 2. Mensaje de error en caso de no estar registrado o que el usuario y la contraseña no coincidan
DESCRIPCIÓN	El administrador puede consultar la lista de fichajes que existen de cada profesor y comprobar qué asistencias han sido registradas	

Tabla 27: caso de uso detallado "Consultar fichajes"

NOMBRE:		Generar informes
ACTORES	Administrador	
PRECONDICIONES	Estar registrado en la base de datos como un usuario administrador	
POSTCONDICIONES		
FLUJO	NORMAL	ALTERNATIVO
	<ol style="list-style-type: none"> 1. Entrar a la aplicación web accediendo mediante usuario y contraseña 2. Generar los informes deseados en la nueva opción que se le muestra al administrador de “Informes” 3. Rellenar los formularios para especificar el tipo de informe 4. Generar y descargar informe 5. Cerrar sesión 	<ol style="list-style-type: none"> 1. Mensaje de error en caso de no estar registrado o que el usuario y la contraseña no coincidan 2. Generación de informe vacío si no hay coincidencias en la consulta requerida
DESCRIPCIÓN	El administrador es el que genera los informes acerca de las asistencias por día o titulación de un profesor o de los profesores a sus clases y tutorías	

Tabla 28: caso de uso detallado "Generar informes"

NOMBRE:		Configurar lector de huellas
ACTORES	Administrador	
PRECONDICIONES	Estar registrado en la base de datos como un usuario administrador	
POSTCONDICIONES		
FLUJO	NORMAL	ALTERNATIVO
	<ol style="list-style-type: none"> 1. Entrar a la aplicación web accediendo mediante usuario y contraseña 2. Comprobar que la conexión con el lector de huellas es la correcta en la opción "Arduino" 3. Rellenar la información en caso de faltar o realizar cambios 4. Cerrar sesión 	<ol style="list-style-type: none"> 1. Mensaje de error en caso de no estar registrado o que el usuario y la contraseña no coincidan 2. Introducción errónea de la información y fallo en la conexión con el lector
DESCRIPCIÓN	El administrador tiene que configurar la conexión con el lector de huellas	

Tabla 29: caso de uso detallado "Configurar lector de huellas"

NOMBRE:		Eliminar huella
ACTORES	Administrador	
PRECONDICIONES	Estar registrado en la base de datos como un usuario administrador	
POSTCONDICIONES		
FLUJO	NORMAL	ALTERNATIVO
	<ol style="list-style-type: none"> 1. Entrar a la aplicación web accediendo mediante usuario y contraseña 2. Seleccionar la opción de "Eliminar huella" 3. Introducir la huella que se desea eliminar y pulsar sobre el botón correspondiente 4. Cerrar sesión 	<ol style="list-style-type: none"> 1. Mensaje de error en caso de no estar registrado o que el usuario y la contraseña no coincidan 2. La huella introducida no existe
DESCRIPCIÓN	El administrador tiene la opción de eliminar huellas registradas	

Tabla 30: caso de uso detallado "Eliminar huella"

NOMBRE:	Registrar usuario	
ACTORES	Administrador	
PRECONDICIONES	Estar registrado en la base de datos como un usuario administrador	
POSTCONDICIONES		
FLUJO	NORMAL	ALTERNATIVO
	<ol style="list-style-type: none"> 1. Entrar a la aplicación web accediendo mediante usuario y contraseña 2. Entrar en la opción de “Registrar” 3. Rellenar el formulario con la información solicitada del nuevo usuario 4. Registrar al usuario 5. Cerrar sesión 	<ol style="list-style-type: none"> 1. Mensaje de error en caso de no estar registrado o que el usuario y la contraseña no coincidan 2. Falta de información o información errónea introducida en el formulario
DESCRIPCIÓN	Los usuarios de tipo administrador son los únicos que pueden crear y registrar a nuevos usuarios	

Tabla 31: caso de uso detallado "Registrar usuario"

4.4. Diagramas de actividades

Un diagrama de flujo o de actividades utiliza rectángulos redondeados para denotar una función específica del sistema, flechas para representar flujo a través de éste, rombos de decisión para ilustrar una ramificación de las decisiones (cada flecha que salga del rombo se etiqueta) y líneas continuas para indicar que están ocurriendo actividades en paralelo. Aquí podéis ver el diagrama de actividades de nuestro sistema basado en un lector de huellas completo cuando un profesor va realizar su primer fichaje de asistencia [44]:

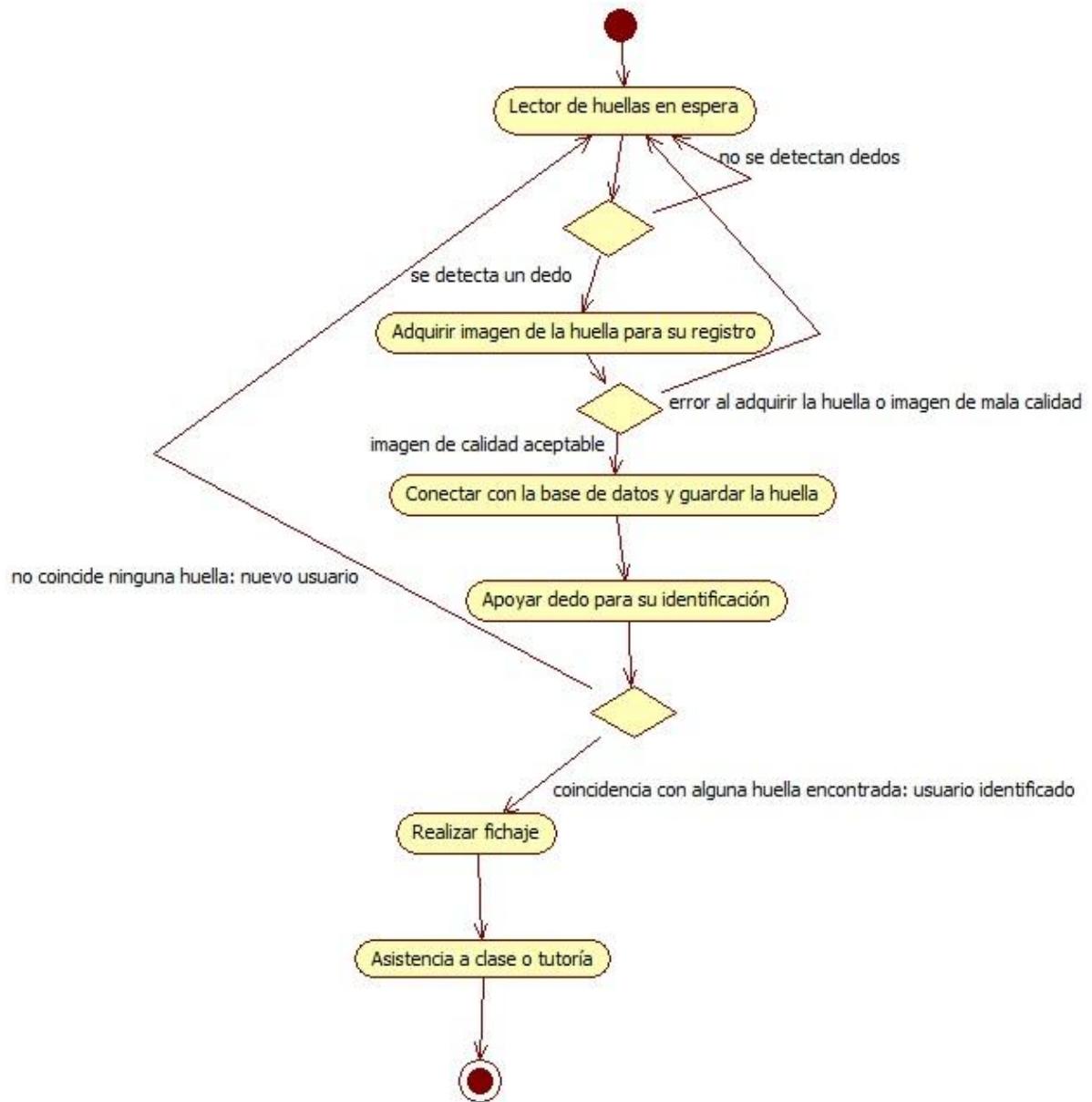


Figura 39: diagrama de actividades

4.5. Cronograma

Una vez recopilados los requisitos del proyecto y cuantificada su dimensión, para la consecución de los objetivos establecidos junto con la redacción del informe / memoria del Trabajo Fin de Grado se ha planteado el siguiente cronograma:

	FEBRERO	MARZO	ABRIL	MAYO	JUNIO
Estudiar y analizar los sistemas hardware de lectura de huella dactilar (25 horas)	25 horas				
Estudiar y analizar los distintos SDK libres y de pago para el desarrollo de aplicaciones basadas en la lectura de huella dactilar (18 horas)	8 horas	10 horas			
Diseño y desarrollo de sistema de fichaje de entrada y salida a clase del profesor (100 horas)		85 horas	15 horas		
Diseño y desarrollo de sistema de fichaje de estudiante a clase (113 horas)			89 horas	24 horas	
Redacción de memoria (44 horas)					44 horas

Tabla 32: planificación estimada

Capítulo 5: Diseño del Sistema

Contenidos

Capítulo 5: Diseño del Sistema.....	56
5.1. Arquitectura del sistema.....	56
5.2. Diseño de la instalación Hardware	58
5.2.1. Arduino	58
5.2.2. Lector de huellas.....	59
5.2.3. Pantalla LCD.....	59
5.2.4. Servidor Raspberry PI 2	60
5.3. Patrón MVC en nuestro proyecto	60
5.3.1. Capa de Presentación.....	61
5.3.2. Capa Lógica.....	62
5.3.3. Capa del Modelo	62
5.4. Sistema de reservas: MRBS	62
5.4.1. Modelo de datos MRBS.....	63
5.4.2. Ampliación y variaciones en el modelo de datos.....	64
5.5. Diseño de la seguridad	68
5.5.1. Seguridad en la identificación biométrica	68
5.5.2. Seguridad en el servidor y en la aplicación web	68
5.5.3. Encriptación AES	69

Este capítulo contiene la arquitectura del sistema que vamos a utilizar en este Trabajo Fin de Grado, tanto las aplicaciones desarrolladas como la parte hardware que se requiere.

5.1. Arquitectura del sistema

En este proyecto se van a desarrollar los siguientes sistemas:

- **Sistema de registro de usuario y de huellas instalado en una aplicación web.**

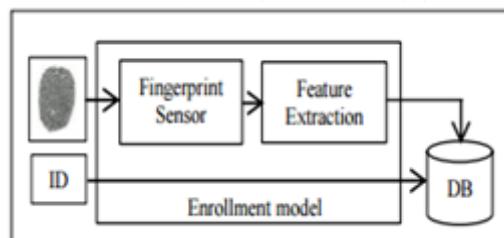


Figura 40: registro de huella [45]

- **Sistema de fichaje instalado en una aplicación web.**

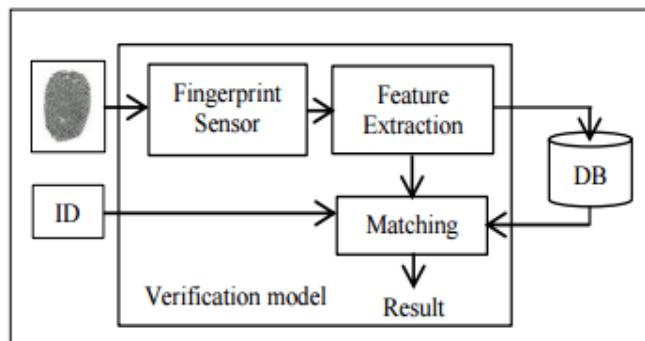


Figura 41: verificación de huella

- **Sistema de control de reservas de aulas con la opción de generación de informes.**

Se seguirá una **arquitectura MVC (Modelo-Vista-Controlador)**, es decir, un patrón de diseño con el que podremos organizar nuestro código en base a su función. Se separa el código en tres capas [46]:

- La **capa del modelo** define la **lógica de negocio**. Representa la información almacenada en una base de datos o en XML, junto con las reglas de negocio que transforman esa información.
- La **vista o capa de presentación** es lo que utilizan los usuarios para interactuar con la aplicación (los gestores de plantillas pertenecen a esta capa). Serán nuestras páginas HTML directamente.
- El **controlador o capa lógica** es un bloque de código que realiza llamadas al modelo para obtener los datos y se los pasa a la vista para que los muestre al usuario. Manipula el modelo y gestiona la vista, es decir, es la parte del código que obtiene dinámicamente datos y genera el contenido HTML.

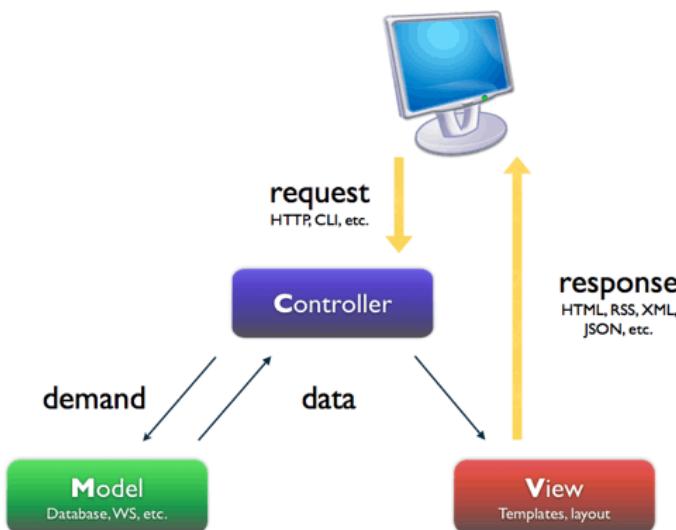


Figura 42: patrón Modelo-Vista-Controlador [47]

5.2. Diseño de la instalación Hardware

Para llevar a cabo el proyecto, necesitamos de ciertos dispositivos hardware con los que posteriormente interactuaremos en la aplicación web que desarrollemos para el sistema.

Aquí vamos a describir cómo se van a satisfacer uno a uno los requisitos de la fase de análisis que nuestro sistema necesita para llevar a cabo el proyecto.

5.2.1. Arduino

Como ya hemos indicado en apartados anteriores, contar con un lector de huellas profesional junto con un SDK compatible se sale de la viabilidad económica que alberga este proyecto. Por ello, se ha decantado por utilizar sistemas de código abierto.

Muestra de ello es **Arduino**, una plataforma de hardware de código abierto, basada en una sencilla placa con entradas y salidas, analógicas y digitales, en un entorno de desarrollo (IDE). Es un dispositivo que conecta el mundo físico con el mundo virtual, o el mundo analógico con el digital [48].

El microcontrolador de la placa se programa usando el **Arduino Programming Language** (basado en **Wiring**) y el **Arduino Development Environment** (basado en **Processing**).

Por lo que básicamente se componen de un microcontrolador y un entorno de desarrollo (IDE), diseñado para facilitar el uso de la electrónica en proyectos multidisciplinarios. Estas placas se crearon con la idea de que cualquier persona pueda realizar proyectos interactivos de forma sencilla, mediante hardware y software muy fáciles de usar y de tal manera que la placa es ampliable [49].

Así, con ella podemos hacer uso de lectores de huellas compatibles con Arduino, el cual emplea el protocolo de comunicación serial, de código abierto y de costes muy reducidos.

5.2.2. Lector de huellas

Una vez contemos con una placa Arduino, sólo nos falta un dispositivo con el que podamos extender este hardware y que tenga la capacidad de leer o escanear huellas dactilares, y que permita su conexión con el microcontrolador.

Un ejemplo es el sensor de **Adafruit**, ideal para realizar un sistema capaz de conseguir una protección de datos o de lo que se requiera por medio del análisis de una huella digital. El sistema realiza procesamiento digital de imágenes (**DSP**) interno, además de incluir capacidades de comparación en base de datos y actualización de la misma. El dispositivo funciona con el protocolo serial, por lo que puede ser utilizado con cualquier microcontrolador o tarjeta de desarrollo, en nuestro caso, Arduino. Además, este lector tiene una versión **Grove**, que será el que utilicemos para hacer uso de la **base shield**.

Grove dispone de un conjunto de herramientas modular que facilita el montaje de la parte electrónica en los Arduinos. Una de ellas serían una base shield, placa que transforma los pines de conexiones en una serie de conectores.

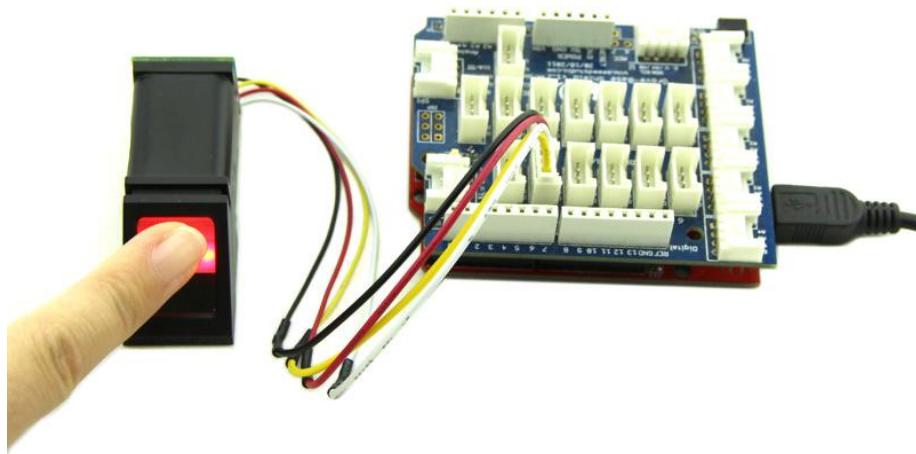


Figura 43: ejemplo de un sensor ADAFRUIT conectado a una base shield Grove de Arduino

5.2.3. Pantalla LCD

Desde el *Starter Kit for Arduino* que indicamos en el apartado 4.1.3 que íbamos a obtener para el desarrollo de proyecto, podemos conseguir una **pantalla de cristal líquido o LCD**. A través de ella ayudaremos dando instrucciones al usuario mientras utiliza el lector de huellas.

5.2.4. Servidor Raspberry Pi 2

También ya indicamos en la viabilidad económica de la sección 4.1.3 que emplearemos una **Raspberry Pi 2** como servidor. Esto es un ordenador de placa reducida (SBC) y de bajo coste, un pequeño dispositivo con el que cualquier persona puede estimular su aprendizaje en cuanto a las ciencias de la computación. Tiene la capacidad de interactuar con el mundo exterior, ofreciendo diversos puertos a los que podremos conectar un monitor, un ratón, un teclado, una memoria USB, Internet, etc. [50].

Tiene muchísimas utilidades, y, entre ellas, puede funcionar como si fuera un servidor de aplicaciones, y ésta es la razón por la que haremos uso de Raspberry Pi. Podremos tener almacenado en ella nuestro sistema de base de datos, tanto del sistema de reservas MRBS como el registro de huellas dactilares.

Al margen de la Raspberry Pi, alojaremos la aplicación web en el servidor web de **Apache** que necesitamos tener instalado en el ordenador local donde se conecte el sensor.

5.3. Patrón MVC en nuestro proyecto

Una vez explicado en qué consiste la arquitectura MVC que vamos a utilizar para nuestro sistema, pasamos a indicar las distintas capas de la arquitectura en nuestro proyecto Fin de Grado.

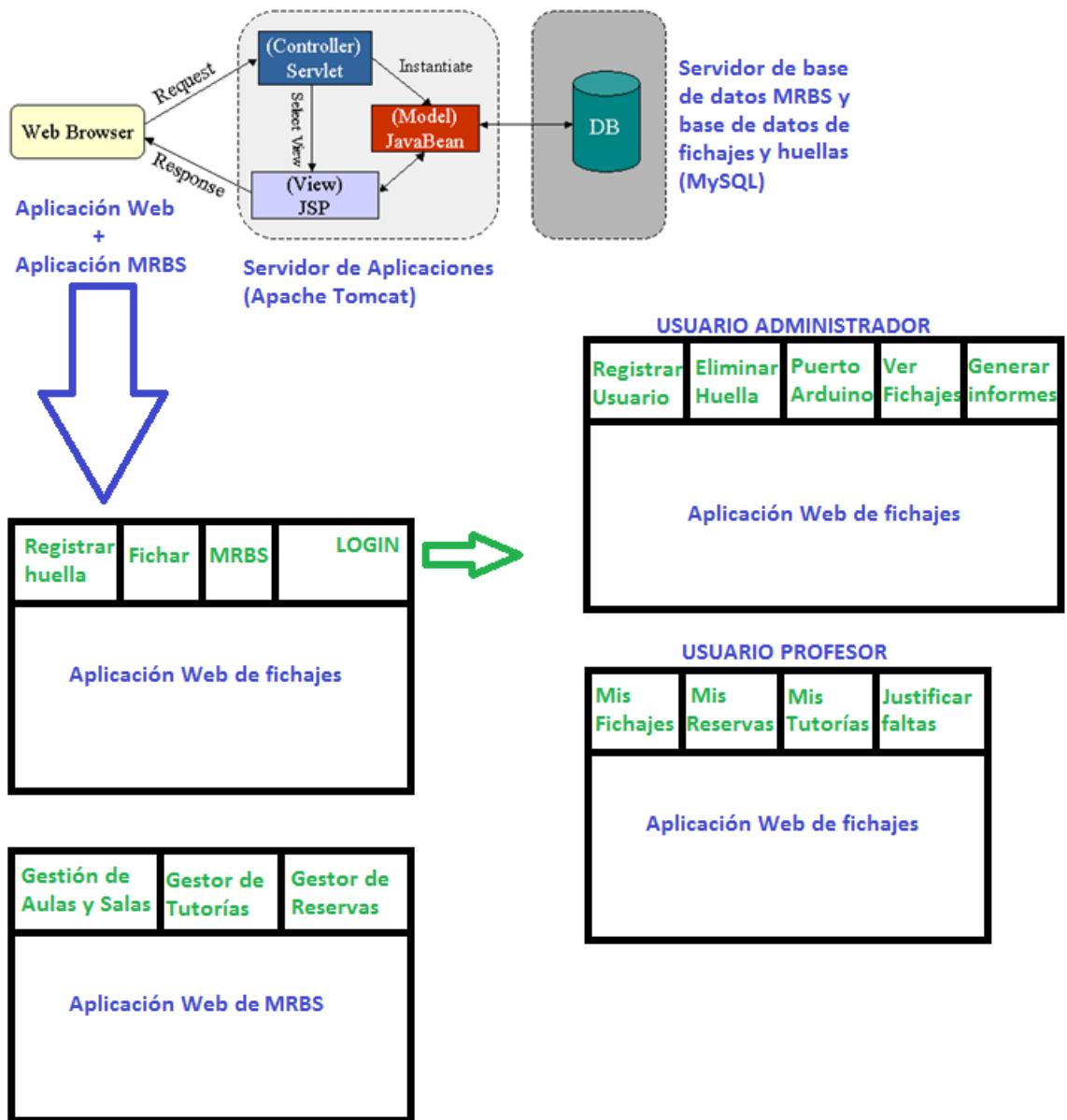


Figura 44: diseño de la aplicación bajo el patrón MVC

5.3.1. Capa de Presentación

Nuestra aplicación web combinada con la aplicación MRBS sería nuestra **Vista**. A través de ella, el usuario o profesor podrá registrar su huella dactilar e identificarse cada vez que asista a alguna clase o a alguna tutoría realizando el correspondiente fichaje. También tendrá la opción de justificar una falta.

El administrador podrá generar distintos informes en PDF por aula y día, por tutorías de un día determinado, por titulación dado un día seleccionado o por tutorías de un profesor específico en un rango de días. De esta manera tendremos la posibilidad de comprobar la asistencia del profesor en unos informes.

El resto de opciones del administrador son: eliminar una huella, cambiar el puerto de conexión del Arduino, consultar la lista completa de fichajes realizados y registrar usuarios.

Además, la aplicación web de reservas se llevará a cabo a través del sistema de reservas MRBS, así como la inserción de tutorías.

5.3.2. Capa Lógica

Nuestro **controlador** será implementado a través de **servlets JSP**, desplegados en el contenedor de servlets **Apache Tomcat**. A su vez, estarán alojados en el servidor web Apache. La aplicación de reservas MRBS estará desarrollada en PHP.

Con esta capa lógica, podremos realizar llamadas a la base de datos MRBS o a la base de datos de fichajes y registros de huellas, para posteriormente pasárselo a nuestra vista, es decir, a nuestra aplicación web descrita en la capa de presentación.

5.3.3. Capa del Modelo

La base de datos MRBS está instalada en **MySQL** en la Raspberry Pi 2 que está funcionando como un servidor web.

Nuestra **lógica de negocio** únicamente consiste en la base de datos MRBS y en la base de datos que almacena las diferentes huellas de los profesores, así como los fichajes, asignaturas, tutorías, etc.

5.4. Sistema de reservas: MRBS

La aplicación **MRBS (Meeting Room Booking System)** [77] es un sencillo sistema de reserva de salas y gestión de horarios. Es idónea para gestionar lugares de reunión, asociaciones, o edificios con salas que necesiten de una reserva. Permite configurar distintos edificios con sus salas independientes, y tiene un sencillo sistema de reservas basado en un visual calendario. Está disponible como servidor y cliente multiplataforma, bajo licencia GPL v2. Dispone de numerosas funcionalidades y características [51]:

- **Completo sistema de reserva de salas.** MRBS permite a los usuarios reservar horarios para un determinado conjunto de salas. Tan solo pinchando en el día requerido, el usuario puede poner gran cantidad de información acerca de la sala que va a reservar.
- **Visualización tipo calendario.** MRBS ofrece una interfaz sencilla para cualquier usuario, mostrando un calendario de reservas con los días disponibles. La vista puede ser de todo el mes, semanal o por día, mostrando en cada caso el calendario para una sala.
- **Definición de varios edificios y salas.** Con MRBS podemos siendo administrador definir varios edificios en los que hacer reserva, cada uno con sus salas independientes. Además se pueden establecer horarios límite para los que hacer reservas e incluso el aforo de las salas.

- **Informes de uso.** Los usuarios administradores también pueden generar informes de uso con una utilidad automática de la aplicación. Pueden exportarse estos informes en formato CSV.
- **Integración de un buscador.**
- **Configuración de distintos niveles de usuario.** Diferentes niveles de autenticación (sólo lectura, usuario o administrador, es decir, nivel 0, 1 y 2 respectivamente) [76].
- **Posibilidad de recibir notificaciones y cambios por correo.**
- **Multilenguaje**, incluyendo el español.
- **Soporte para web/Intranet**, disponible para cualquier estación de trabajo a través de un navegador.

En cuanto a las limitaciones de MRBS, solo cabe indicar que ciertas configuraciones simples como incluir un logo o añadir un título al sistema de reserva, están fuera de la interfaz de usuario y se requiere realizar manualmente en los ficheros de configuración.

Por otro lado, MRBS tiene ciertos requisitos en el sistema para funcionar (recordamos que es una aplicación multiplataforma) [51]:

- A nivel de servidor, solo se necesita tener soporte para base de datos **MySQL** o **PostgreSQL**, y contar con un **servidor web** (por ejemplo, *Apache*) con soporte para **PHP** (a partir de la versión 5).
- A nivel de cliente, MRBS puede ser accedido desde cualquier plataforma o sistema operativo: tan solo hace falta **conexión a la red** apropiada y un **navegador** de internet.

Tras esta breve explicación acerca de MRBS, podemos concluir que esta aplicación nos será muy útil para diseñar nuestro sistema de reservas. Analicemos ahora los componentes de esta aplicación web:

5.4.1. Modelo de datos MRBS

Partiendo del Trabajo Fin de Grado de **Jorge Alvarado Díaz**, que a su vez está basado en el proyecto “**eCUM: Control de Asistencia**” de **Milagros Membrillo Jiménez**, obtenemos la base de datos de MRBS del primero, pues contempla la inserción de tutorías y fichaje.

La base de datos del sistema eCUM es la usada por el sistema de reservas del Centro Universitario de Mérida, y con ella podemos aprovecharnos de la información que se publica en tiempo real sobre las reservas y ocupación de las aulas/salas.

MRBS se compone de las siguientes tablas, que no se pueden alterar:

- **mrbs_area**: tabla que almacena las diferentes áreas que contienen las salas disponibles.
- **mrbs_entry**: es una de las tablas más importantes de MRBS al ser dónde se guardan las reservas realizadas y los datos de las mismas.

- **mrbs_repeat:** aquí se tienen las opciones de las reservas que se repiten: en MRBS se puede realizar reservas que se dupliquen en un espacio de tiempo como puede ser cada día, cada semana, cada mes, etc.
- **mrbs_room:** se almacenan las diferentes salas con sus datos.
- **mrbs_users:** en caso de usar la autenticación basada en base de datos aquí se almacenarían los diferentes usuarios del sistema.
- **mrbs_variables:** variables del sistema MRBS.
- **mrbs_zoneinfo:** tabla dónde se almacenan las zonas horarias soportadas por el sistema.

La ampliación realizada en el trabajo de Milagros Membrillo Jiménez consistía en añadir nuevas tablas a la base de datos MRBS sin interferir en las originales. Incluyó:

- **mrbs_usuario:** en esta tabla se almacenan los usuarios / profesores de la aplicación de fichaje realizada en ese trabajo.
- **mrbs_claves:** es la tabla encargada de vincular los usuarios con sus claves.
- **mrbs_asignatura:** tabla dónde se almacenan las diferentes asignaturas impartidas por los profesores.
- **mrbs_asigprof:** se utiliza para relacionar un profesor con las asignaturas que imparte.
- **mrbs_fichaje:** se encarga de guardar los fichajes realizados por un profesor conectándolos con las respectivas reservas del sistema MRBS.

Finalmente, Jorge Alvarado Díaz incluyó un par de variaciones en las tablas mrbs_usuario (amplió el número de atributos de esta tabla para tener más información sobre los profesores) y mrbs_asignatura (cambió la longitud máxima del atributo "nombre"), además de añadir las dos siguientes tablas para poder almacenar también los horarios de tutorías del profesorado y los diferentes períodos lectivos:

- **mrbs_periodo:** en esta tabla se almacenan los intervalos de fechas dónde están vigentes los diferentes períodos lectivos (primer semestre, segundo semestre, periodo lectivo de exámenes y periodo no lectivo).
- **mrbs_tutoria:** contiene los horarios de tutorías de los diferentes profesores.

5.4.2. Ampliación y variaciones en el modelo de datos

En nuestro caso, tendremos que realizar varias modificaciones, sobre las tablas **mrbs_claves**, **mrbs_asigprof** y **mrbs_fichaje**, así como la creación de la tabla **mrbs_titulacion**.

La tabla de **claves** almacenará los identificadores de las huellas y relacionará la huella con su usuario. Es decir, simplemente tendrá tres columnas:

- **id.** Identificará a la fila de manera única.
- **id_usuario.** Contendrá el identificador del usuario al que pertenece la huella en cuestión.

- **id_huella.** Almacena el identificador de la huella con el que podemos referenciar en el lector a una huella. Éste podrá ir de 1 a 260 (en el caso de nuestro lector, que no permite guardar más huellas), y los identificadores son únicos.

En la tabla de **fichajes** añadiremos los tres siguientes campos:

- **confianza.** Este campo contiene un parámetro numérico el cual especifica el porcentaje de fiabilidad con el que se ha obtenido la huella que coincide con la ofrecida en el fichaje. Es decir, nos proporciona la confianza con la que el usuario presente es el mismo al usuario que ha encontrado en la base de datos de huellas.
- **entrada_salida.** Campo que podrá ser “entrada” o “salida”, dependiendo del tipo de fichaje que realice el profesor: llega a clase o sale de ella.
- **tutoria.** Clave externa para conocer la tutoría a la que se refiere el fichaje del profesor. En el caso de que el fichaje sea hacia una clase o aula, se referenciará mediante **identry** (identificador del aula reservada).

Por último, la tabla que relaciona la asignatura con los profesores que la imparten (**mrbs_asigprof**) tendrá también un campo donde se indique la titulación:

- **id_titulacion.** Este campo hará referencia a la titulación a la que pertenece la asignatura, a través del identificador de la titulación (**mrbs_titulacion**).
 - La tabla **mrbs_titulacion** contiene los dos siguientes campos: **id** (clave primaria) y **nombre**.

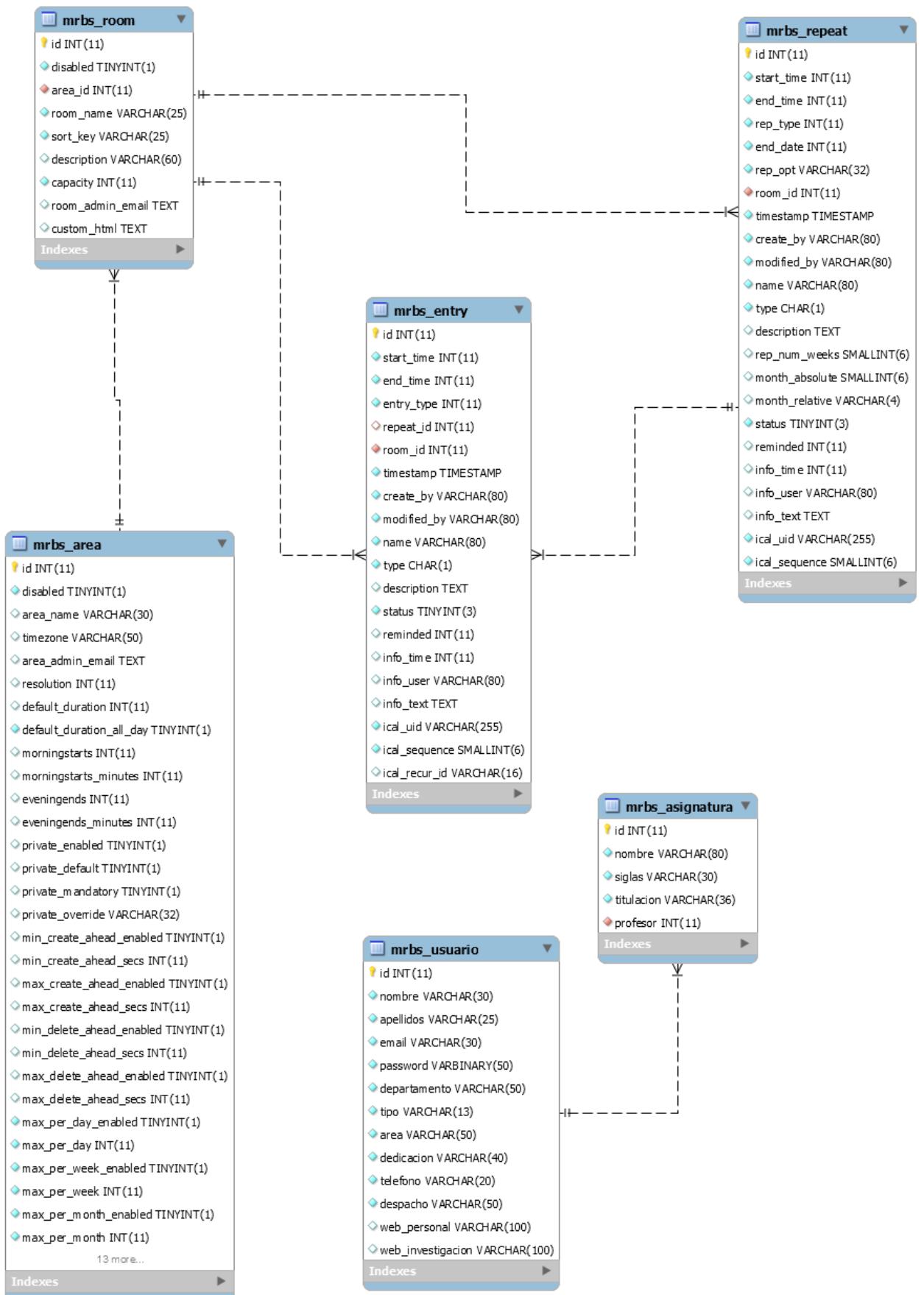


Figura 45: diagrama de la base de datos (parte 1)

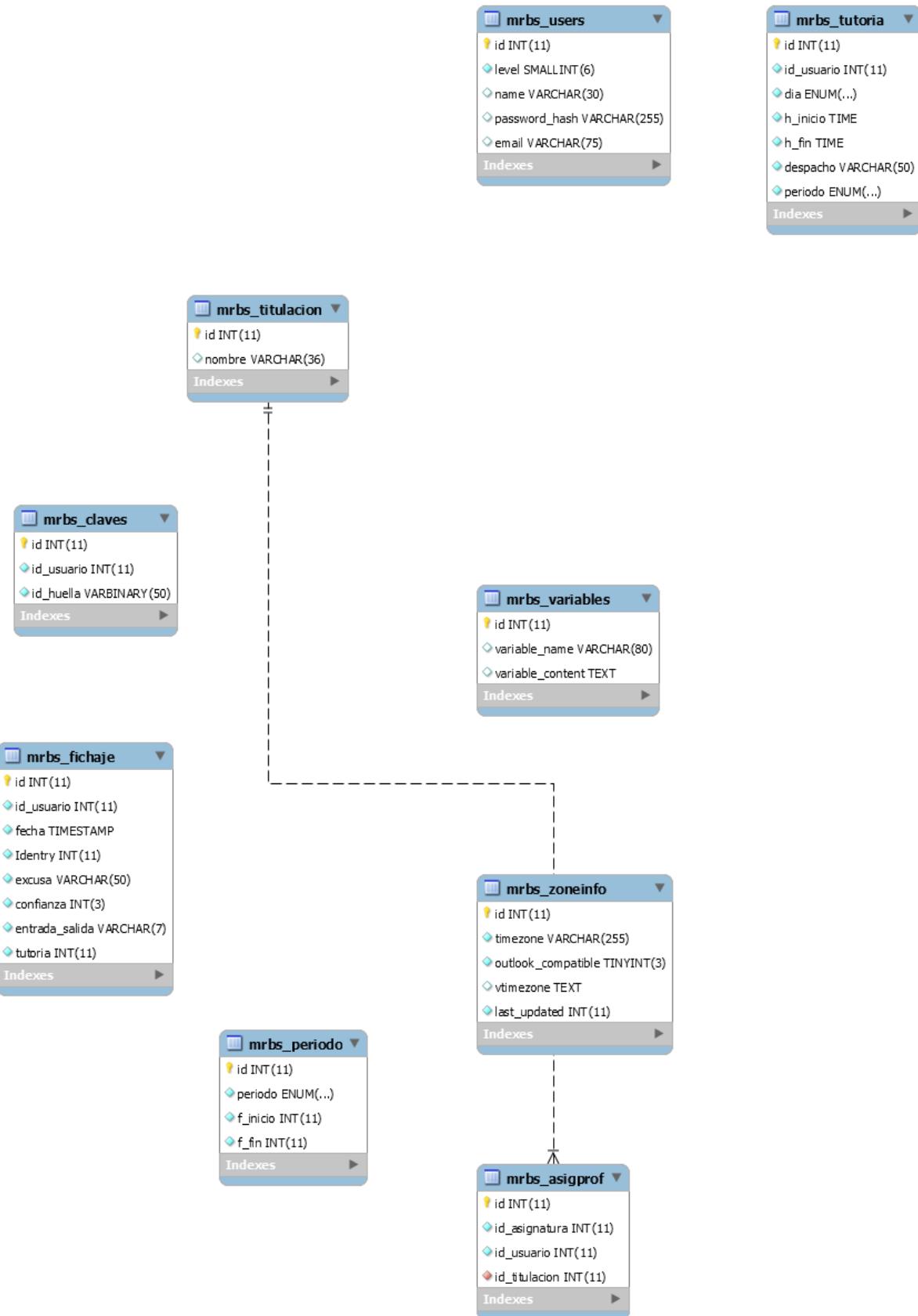


Figura 46: diagrama de la base de datos (parte 2)

5.5. Diseño de la seguridad

En este apartado vamos a explicar la seguridad que va a tener nuestro proyecto en cada uno de sus componentes: lector de huellas, aplicación web y servidor.

5.5.1. Seguridad en la identificación biométrica

El lector de huellas de Adafruit ofrece **cinco niveles** o grados de seguridad (del 1 al 5). La **tasa de falsa aceptación** (FAR) es menor de 0.001% en nivel de seguridad 3, que es el nivel por defecto. La **tasa de falso rechazo** (FRR) es menor al 1%. A menor nivel de seguridad (1), mayor será el FAR y menor el FRR, y a mayor nivel de seguridad (5), viceversa.

Con el sensor, reemplazaremos para realizar los fichajes la autenticación típica y vulnerable de usuario y contraseña del profesor, o la identificación a través de tarjetas NFC, y otorgaremos así la seguridad que ofrece la utilización de un sistema biométrico de este tipo (lector de huellas dactilares) [52].

De acuerdo con el número de características obtenidas, se elige si la imagen de la huella es aceptable o no para proceder a almacenar su huella. Cuando se quiere identificar a un usuario, el sensor busca coincidencias entre las huellas registradas, y si se encuentra una huella con una alta confianza, el lector devuelve el identificador y se procede a la identificación. En este punto, se realiza la conexión a la base de datos y se llama al servidor MySQL para buscar al usuario correspondiente por medio del identificador [53].

5.5.2. Seguridad en el servidor y en la aplicación web

A continuación se aborda los problemas clásicos de seguridad en una base de datos.

En el servidor se encuentra almacenada la base de datos, y esta cuenta con **Autentificación**, ya que los usuarios deben acceder mediante su correo electrónico y contraseña para poder ejercer algún tipo de acción sobre ella. De esta forma, se evita que personas que no estén registradas puedan acceder al sistema.

Además, existe el concepto de **Autorización**, pues tenemos dos niveles de usuario: profesor (solo puede realizar consultas) y administrador. Cada usuario sólo puede acceder a la información sobre la que tiene permisos.

En cuanto a la **Privacidad**, las contraseñas y los identificadores de las huellas almacenadas se guardan de forma cifrada por medio de la **encriptación AES** (explicada más adelante), y se oculta así la información que puede ser sensible a que terceras personas la consulten con total libertad y puedan tener acceso a datos sobre los cuales no tienen permiso.

Por último, la **Integridad** es una característica de la que se encarga de garantizar que una entidad (fila o registro) siempre se relacione con otras entidades válidas, existentes en la base de datos. Eso implica que en todo momento dichos datos sean correctos, sin repeticiones innecesarias, datos perdidos y relaciones mal resueltas. Estas funciones las lleva a cabo el propio sistema gestor de base de datos [54].

5.5.3. Encriptación AES

Para proteger las contraseñas de los usuarios y los identificadores de las huellas que se almacenan, se va a encriptar esta información para que no pueda ser consultada. Al utilizar el sistema de gestión de bases de datos MySQL, tenemos la oportunidad de utilizar diferentes **cifrados** (SHA, MD5, AES, HAVAL...) [55].

Finalmente nos decantamos por la **encriptación AES**, el algoritmo más completo y complejo que ofrece **MySQL** el cual tiene la posibilidad de revertirse y se utiliza con una **clave privada**, es decir, debemos proporcionar dicha clave para encriptar y desencriptar los datos. Únicamente sería vulnerable en el caso de que alguien pudiera acceder al código fuente de nuestra aplicación y consiguiera la clave que se está utilizando para encriptar. Las contraseñas y los identificadores de las huellas serán campos de **tipo BLOB**, pues el resultado de la operación será un dato binario muy aleatorio [56].

La encriptación AES es un algoritmo de cifrado de **clave simétrica** (método criptográfico en el cual se usa una misma clave para cifrar y descifrar mensajes en el emisor y el receptor), específicamente por bloques, de los más usados y populares en este tipo de criptografía. No es vulnerable al criptoanálisis diferencial y lineal, siendo necesario una gran cantidad de textos encriptados y gran procesamiento para su análisis. Según los investigadores, solo se ha registrado un único ataque con éxito, en 2011.

AES tiene un tamaño de bloque fijo de 128 bits, y el tamaño de clave puede ser de 128, 192 o 256 bits. Opera en una matriz de estado, de 4x4 bytes, y el cifrado se realiza con el operador “*exclusive OR*” (**XOR**) [57] [58].

Capítulo 6: Implementación

Contenidos

Capítulo 6: Implementación.....	70
6.1. Análisis y diseño de la arquitectura	70
6.1.1. Capa de Presentación.....	70
6.1.2. Capa lógica.....	73
6.1.3. Capa del modelo	84
6.2. Arquitectura Hardware	86
6.2.1. Programa Arduino.....	87
6.2.2. Servidor Raspberry Pi 2	89
6.3. Implementación de la seguridad	90
6.3.1. Implementación de seguridad en la identificación biométrica.....	90
6.3.2. Implementación de seguridad en el servidor y en la aplicación web....	90

En el capítulo anterior explicamos el diseño del sistema, mencionando las herramientas y tecnologías que íbamos a usar para la realización del trabajo. Es el momento de hablar de la implementación del mismo, donde desarrollaremos las tecnologías utilizadas.

6.1. Análisis y diseño de la arquitectura

6.1.1. Capa de Presentación

El servicio web se ha creado a partir del lenguaje de programación **Java**, construyendo el sitio web mediante las tecnologías de **JSP**, **HTML 5**, **CSS 3**, **Bootstrap** y **JQuery**, y teniendo en cuenta que los informes se han llevado a cabo mediante **Jasper Reports**. Damos paso a la explicación de estos lenguajes de programación utilizados:

- **JSP (Java Server Pages).** Tecnología orientada a crear páginas web con programación en Java. Mediante JSP podemos crear aplicaciones web que se ejecuten en variados servidores web, de múltiples plataformas, ya que Java es en esencia un lenguaje multiplataforma. Las páginas JSP están compuestas de

código HTML/XML mezclado con etiquetas especiales para programar scripts de servidor en sintaxis Java.

El motor de las páginas JSP está basado en los servlets de Java, es decir, programas en Java destinados a ejecutarse en el servidor. Para poder desplegar y correr páginas JSP es necesario un contenedor de servlet.

En JSP creamos páginas con extensión .jsp que incluyen, dentro de la estructura de etiquetas HTML, las sentencias Java a ejecutar en el servidor [74].

- **HTML 5 (HyperText Markup Language, versión 5).** Es la última versión de HTML, el lenguaje básico de la *World Wide Web*. Se trata de una nueva versión con más elementos, atributos y comportamientos y facilitan y simplifican el trabajo del programador web. Contiene un conjunto más amplio de tecnologías que permite además a los sitios web y a las aplicaciones ser más diversas y de gran alcance [75].
- **CSS 3 (cascading style sheets, hoja de estilo en cascada versión 3).** Es un lenguaje usado para definir y crear la presentación de un documento estructurado escrito en HTML o XML2 (y por extensión en XHTML). La tercera versión, CSS3, está dividida en varios documentos separados, llamados "módulos". Cada módulo añade nuevas funcionalidades a las definidas en CSS2, de manera que se preservan las anteriores para mantener la compatibilidad [59].
- **JQuery.** Es una biblioteca de JavaScript. Permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web [60].
 - **JavaScript.** Lenguaje de programación interpretado, orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Se utiliza principalmente en su forma del lado del cliente, implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas aunque existe una forma de JavaScript del lado del servidor [61].
- **Bootstrap.** Es un framework o conjunto de herramientas de código abierto desarrollado por Twitter, útil para el diseño de sitios y aplicaciones web. Es el *framework* más popular que combina HTML, CSS y JavaScript para desarrollar sitios *responsive*, es decir, adaptables para cualquier tipo de plataforma (Tablet, PC, Smartphone...). Entre las extensiones que ofrece se encuentran plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y más extensiones de JavaScript opcionales. Al usar esta tecnología, nuestra aplicación se puede utilizar en cualquier plataforma [62].

- **JasperReports Server.** Biblioteca de creación y análisis de informes integrable en cualquier aplicación, sea web, móvil o hasta formato para impresora. Se generan informes en multitudes formatos: PDF, HTML, XLS, CSV y XML. Está escrito completamente en Java [63].

```
<%@include file="includes/menuProfesor.jsp"%>
<h2>Sistema de control de asistencia - Profesor</h2>
<div class="container">
    <div class="row">
        <div class="col-md-6">
            <h2>Consultar mis fichajes</h2>
            
            <p>Puedes consultar tus fichajes a clase o a tutorías realizados en la sección correspondiente</p>
        </div>
        <div class="col-md-6">
            <h2>Consultar mis reservas</h2>
            
            <p>Las reservas que hayas realizado puedes mostrarlas en el apartado de "Reservas" </p>
        </div>
    </div>
    <div class="row">
        <div class="col-md-6">
            <h2>Consultar mis tutorías</h2>
            
            <p>La opción del menú "Tutorías" proporciona su lista de horas tutorizadas</p>
        </div>
        <div class="col-md-6">
            <h2>Justificar falta de asistencia</h2>
            
            <p>Si algún día de clase o tutorías ha faltado, puede justificar la falta en la sección "Justificar falta" </p>
        </div>
    </div>
</div>
```

Algoritmo 1: ejemplo de código en una página JSP donde se refleja el uso de JSP, HTML y Bootstrap

Partiendo de la existencia de un **administrador** ya creado de manera manual en la base de datos, el mismo administrador ya podría acceder al sitio web con sus credenciales y donde podrá realizar lo siguiente:

- Crear más administradores y registrar a los profesores
- Generar los diferentes informes
- Consultar los fichajes de todos los profesores
- Eliminar huellas almacenadas
- Configurar el puerto de conexión con Arduino

Desde el inicio de la web, una vez se haya registrado el **profesor**, con su correo y contraseña podrá almacenar su huella dactilar, así como realizar el fichaje correspondiente.

Si se identifica como usuario haciendo “*Login*”, el profesor tiene diferentes opciones de consulta:

- Ver sus fichajes
- Ver sus reservas de aulas
- Ver sus horarios de tutorías
- Justificar una falta de asistencia

Todas estas funcionalidades que ofrece la aplicación vienen explicadas cómo llevarlas a cabo en las primeras páginas de inicio de la web.

Por otro lado, tenemos la aplicación de reservas MRBS junto a la gestión de tutorías incorporada por el proyecto de Jorge Alvarado. Como ya dijimos en la fase de diseño (5.4), MRBS es una aplicación útil para la gestión de reserva de salas y gestión de horarios basada en PHP.

6.1.2. Capa lógica

Nuestro servidor de aplicaciones será el contenedor de servlets **Tomcat** [64], que a su vez se aloja en el servidor web **Apache**. Nuestro controlador será implementado a través de **servlets JSP**. En definitiva, utilizaremos el contenedor web Apache Tomcat a partir de su versión 7.0, que al incluir el compilador **Jasper** puede compilar JSPs convirtiéndolos en servlets.

Por otro lado, la aplicación MRBS no ha sufrido modificación alguna y es independiente de la aplicación de fichajes, por lo que repetimos que su desarrollo está en el lenguaje PHP, y se encuentra desplegada en el servidor web de Apache con la versión 2.4.10.

Para desarrollar la aplicación se ha utilizado **Eclipse** [65], una plataforma de software compuesta por un conjunto de herramientas de programación de código abierto y multiplataforma.

Vamos a describir la estructura de esta capa intermedia, es decir la arquitectura de la lógica, que se compone a su vez de controladores, servicios y vista [66].

Los **controladores** son los que reaccionan a la petición del cliente para posteriormente ejecutar la acción adecuada y crear el modelo pertinente. Cada vista solo puede ser asociada a un único controlador, por lo que se ha de tener una variable de tipo “*controller*” que notifique a la vista cuál es su controlador o modelo asignado. De igual manera, el controlador tiene una variable llamada “*View*” que apunta a la **vista**. Así, pueden enviarse mensajes directos el uno al otro y al mismo tiempo, a su **modelo**.

En nuestra aplicación, para ejecutar la acción adecuada se parte del servlet “*Action.java*”, el cual da paso al correspondiente controlador dependiendo de la acción seleccionada:

- **Action.java**. Este servlet es el punto de partida en los controladores. Para ello contiene los condicionales para acceder al controller que se seleccione.

Extiende de la clase HttpServlet, del paquete javax.servlet.http. Se refleja en la URL la palabra “Action” mediante la anotación @WebServlet("/Action"), por lo que la sintaxis de las direcciones son:
<http://URLservidor/TFG>Action>.

```
@WebServlet("/Action")
public class Action extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public void processRequest(HttpServletRequest req, HttpServletResponse res)
        throws IOException, ServletException {
        String accion = req.getParameter("action");
        if (accion.contains("Usuario")) {
            req.getRequestDispatcher("/UsuarioController").forward(req, res);
        } else if (accion.contains("Huella")) {
            req.getRequestDispatcher("/HuellaController").forward(req, res);
        } else if (accion.contains("Admin")){
            req.getRequestDispatcher("/AdminController").forward(req, res);
        } else if (accion.contains("Profesor")){
            req.getRequestDispatcher("/ProfesorController").forward(req, res);
        } else if (accion.contains("informe")){
            req.getRequestDispatcher("/InformeController").forward(req, res);
        }
    }
}
```

Algoritmo 2: código del servlet Action

- AdminController. Servlet que alberga cada una de las peticiones que tienen que ver sobre las opciones disponibles para un usuario administrador. Este controlador recibe las peticiones del Action por medio de “AdminController” y decide qué acción ejecutar entre las siguientes:
 - cambiarPuerto(). Llama al método correspondiente para poder realizar el cambio del nombre del puerto de conexión del Arduino.
 - listaHuellas(). Muestra una lista de las huellas existentes en la base de datos para la página en la que el administrador puede eliminar una huella.
 - Aula_Dia(). Método que permite embeber la lista desplegable en el informe de aulas por día, para que así el administrador pueda seleccionar un aula existente.
 - Titulacion_Dia(). Método que permite embeber la lista desplegable en el informe de una titulación por día, para que así el administrador pueda seleccionar la titulación sobre la que quiera generar el informe.
 - Tutoria_Profesor(). Método que permite embeber la lista desplegable en el informe de las tutorías de un profesor, para que así el administrador pueda seleccionar al profesor sobre el que quiere generar el informe.
 - lista_Fichajes(). Ofrece la lista completa de los fichajes realizados por todos los profesores.

```

private void Aula_Dia(HttpServletRequest req, HttpServletResponse res)
    throws IOException, ServletException {
try {
    InformesService aulas = new InformesServiceBD();
    List<Aula_Titulacion> result = aulas.getAulas();
    if (result == null) {
        res.sendRedirect("inicio.jsp?error=No existen aulas");
        return;
    } else {
        req.setAttribute("lista", result);
        req.getRequestDispatcher("aula_dia.jsp").forward(req, res);
    }
} catch (SQLException e2) {
    res.sendRedirect("inicio.jsp?error=Ha ocurrido un error en la base de datos");
}
}

```

Algoritmo 3: ejemplo de código en el AdminController (método para el informe de aula por días)

- **HuellaController.** En este controlador se llevan a cabo todas las acciones relacionadas con el lector de huellas, a través de las peticiones en el Action llamadas “HuellaController”.
 - Fichaje(). Se encarga de conectar con el Arduino para ejecutar el programa donde el lector de huellas verifica la identidad del profesor y así realizar el correspondiente fichaje según la hora en la que lo efectúe.
 - registrarHuella(). Una vez que se identifica un profesor, se ejecuta el programa del Arduino donde el lector de huellas almacena una huella nueva a la base de datos.
 - eliminarHuella(). Ofrece al administrador la posibilidad de eliminar huellas dactilares que ya se encuentren almacenadas.
 - PrimeraConexionArduino(). Establece la primera conexión con el Arduino y comprueba que no se abren más conexiones con el mismo. Se pasa un tiempo “time” con el que la página se queda esperando para que el usuario pueda interactuar con el sensor.
 - reconnectarArduino(). Detiene la conexión con el Arduino cada vez que se efectúa una acción para que el lector no se quede funcionando mientras no se utilice.
 - realizarFichaje(). Mediante este método se busca si el fichaje realizado es correcto y lo relaciona con la clase o tutoría correspondiente a la hora en la que ficha.

```

public void eliminarHuella(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {
    try {
        String id = req.getParameter("id");
        sensor.setAccion(id);
        if (primeraConexionArduino(7000)) {
            if (datosFichaje == operacionFallida) {
                if (reconectarArduino(7000)){
                    res.sendRedirect("inicio.jsp?error=Ha ocurrido un problema con la conexión con Arduino. Por favor, revise la conexión con Arduino y con el servidor");
                } else {
                    res.sendRedirect("inicio.jsp?error=El tiempo para su identificación ha expirado. Intentalo de nuevo");
                }
            }
            return;
        } else if (datosFichaje == primeraConexion) {
            res.sendRedirect("inicio.jsp?error=Se ha establecido la conexión con Arduino por primera vez. Por favor, vuelva a intentarlo");
            return;
        } else if (datosFichaje == errorConexion) {
            res.sendRedirect("inicio.jsp?error=Se ha producido un error de conexión con el Arduino. Por favor, compruebe que está bien conectado al PC");
            return;
        } else {
            ClaveService clave = new ClaveServiceBD();
            boolean eliminado = clave.eliminarHuella(Integer.parseInt(id));
            if (eliminado) {
                res.sendRedirect("inicio.jsp?mensaje=La huella ha sido eliminada correctamente");
            } else {
                res.sendRedirect("inicio.jsp?error=Ha ocurrido un error en la base de datos al eliminar la huella");
            }
        }
    } else {
        reconectarArduino(7000);
        res.sendRedirect("inicio.jsp?error=Ha ocurrido un problema con la conexión con Arduino. Por favor, revise su conexión y vuelva a intentarlo");
        return;
    }
} catch (Exception e2) {
    res.sendRedirect("inicio.jsp?error=Ha ocurrido un fallo al eliminar la huella");
}
}

```

Algoritmo 4: ejemplo de código en el HuellaController (método para eliminar una huella)

- **InformeController.** A partir de este servlet, al que se llega mediante la acción “*InformeController*”, el administrador puede generar los cuatro diferentes informes.
 - *AulaDia()* . Método que se corresponde y permite realizar el informe de aulas por día.
 - *TitulacionDia()* . Método que se corresponde y permite realizar el informe de una titulación por día.
 - *TutoriaDia()* . Método que se corresponde y permite realizar el informe de las tutorías en un día determinado.
 - *Tutoria_Profesor()* . Método que se corresponde y permite realizar el informe de las tutorías de un profesor.

```

private void TutoriaProfesor(HttpServletRequest req, HttpServletResponse res) throws IOException, ServletException {
    List<String> parametros = new ArrayList<String>();
    parametros.add(req.getParameter("diaInicio"));
    parametros.add(req.getParameter("diaFinal"));
    parametros.add(req.getParameter("profesor"));
    req.setAttribute("parametros", parametros);
    req.getRequestDispatcher("informe_TutoriaProfesor.jsp").forward(req, res);
}

```

Algoritmo 5: ejemplo de código en el InformeController (método para el reporte de tutorías de un profesor)

- **ProfesorController.** El servlet que recibe las llamadas de “*ProfesorController*” efectúa las siguientes posibles acciones:
 - *Fichajes()* . Muestra los fichajes correspondientes del profesor que está identificado.
 - *Reservas()* . Muestra las reservas de aulas correspondientes del profesor que está identificado.

- `Tutorias()`. Muestra la lista de tutorías correspondientes del profesor que está identificado.
- `Faltas()`. Recupera las excusas de cada fichaje para que el profesor elija cuál falta quiere justificar.
- `justificarFalta()`. Cambia la excusa de un fichaje para que así una determinada falta de un profesor pueda ser justificada, además de establecer la clase o tutoría a la que se refiere la falta.

```
private void tutorias(HttpServletRequest req, HttpServletResponse res) throws IOException, ServletException {
    HttpSession session = req.getSession();
    String id = (String) session.getAttribute("id");
    try {
        TutoriaService misTutorias = new TutoriaServiceBD();
        List<Tutoria> result = misTutorias.verTutorias(id);
        if (result==null) {
            res.sendRedirect("inicio.jsp?error=No tienes ninguna tutoría registrada");
            return;
        } else {
            req.setAttribute("lista", result);
            req.getRequestDispatcher("tutorias.jsp").forward(req, res);
        }
    } catch (SQLException e2) {
        res.sendRedirect("inicio.jsp?error=Ha ocurrido un error en la base de datos");
    }
}
```

Algoritmo 6: ejemplo de código en el ProfesorController (método para mostrar todas sus tutorías)

- **UsuarioController.** Servlet que recibe las peticiones del Action a través de “*UsuarioController*”, y selecciona una de estas dos acciones posibles:
 - `Login()`. Identifica a un usuario por medio de su correo electrónico y su contraseña, y permite acceder a las opciones correspondientes según sea un profesor o un administrador.
 - `registrarUsuario()`. Guarda la información de un nuevo usuario que registre un administrador.

```

public void Login(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException {
    String email = req.getParameter("emailLogin");
    String password = req.getParameter("passwordLogin");
    try {
        UsuarioService userService = new UsuarioServiceBD();
        boolean acceso = userService.login(email, password);
        if (acceso) {
            Usuario user = userService.getUserByEmail(email);
            HttpSession session = req.getSession(true);
            session.setAttribute("id", String.valueOf(user.getId()));
            session.setAttribute("tipo", user.getTipo());
            session.setAttribute("Nombre", user.getNombre());
            session.setAttribute("Apellidos", user.getApellidos());
            session.setAttribute("email", user.getEmail());

            req.getRequestDispatcher("inicio.jsp").forward(req, res);
        } else {
            res.sendRedirect("login.jsp?error=Error en el usuario y/o la password");
            return;
        }
    } catch (Exception e2) {
        System.out.println(e2.toString());
        res.sendRedirect("login.jsp?error=Ha ocurrido un fallo en la base de datos");
    }
}

```

Algoritmo 7: ejemplo de código en el UsuarioController (método con el que se valida a un usuario que intenta acceder con sus credenciales)

Para consultar la base de datos desde esta capa lógica, se hace uso de los llamados **servicios** o *services*. Son clases que efectúan la operación requerida por el controlador, y lo realiza obteniendo los datos necesarios de la base de datos a la que se conecta mediante un pool de conexiones, y a través del conector **JDBC** (Java Database Connectivity) [70]. Una vez conseguidos los datos, se modelan dependiendo de la capa de modelo para que la información sea devuelta a la vista. A continuación os describimos la serie de servicios que se han utilizado:

- **ClaveService**. Contiene los métodos relacionados con las claves con la que el profesor realiza los fichajes, es decir, las operaciones que tienen que ver con las huellas:
 - `getUserByFingerprint (int huella)` . Obtiene a un usuario registrado en la base de datos a partir del identificador de su huella dactilar, y devuelve toda la información de dicho usuario.
 - `registrarHuella (Clave clave)` . Inserta en la base de datos una nueva clave al haber registrado la huella un usuario. Se pasa como parámetro un objeto “clave” que incluye el identificador del usuario y de la huella, y, si todo funcionó correctamente en la iteración con la base de datos, devuelve un booleano a verdadero, en caso contrario, a falso.
 - `listaHuellas ()` . Devuelve la lista completa de huellas almacenadas en un “ArrayList” de claves, útil posteriormente para informar en la página donde el administrador puede eliminar huellas.

- o `eliminarHuella (int huella)`. Realiza un borrado en la tabla de claves de los datos relacionados con la huella especificada. Devuelve un booleano dependiendo de si la operación se pudo llevar a cabo o no.

```

public boolean registrarHuella(Clave clave) {
    ResultSet resultados = null;
    if (getUserByFingerPrint(clave.getHuella()) != null) {
        return false;
    }
    try {
        Statement sentencia = ConexionUtil.openConnection();
        synchronized (sentencia) {
            String query = "INSERT INTO mrbs_claves (id_usuario, id_huella) "
                + "VALUES ('"
                + clave.getUsuario()
                + "', AES_ENCRYPT('"
                + clave.getHuella() + "', '██████████'))";
            sentencia.executeUpdate(query);
        }
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        // Se cierra resultSet
        if (resultados != null) {
            try {
                resultados.close();
            } catch (SQLException ex) {
                Logger.getLogger(ClaveServiceBD.class.getName()).log(
                    Level.SEVERE, "No se pudo cerrar el Resulset", ex);
            }
        }
    }
    return true;
}

```

Algoritmo 8: ejemplo de código en el ClaveService (método para guardar una huella)

- **FichajeService**. Contiene los métodos relacionados con los fichajes que puede realizar los profesores en sus clases o tutorías.
 - o `fichar (Fichaje ficha)`. Inserta los datos correspondientes en la tabla de fichajes cuando se materializa uno de estos. Devuelve un booleano dependiendo de si la operación se pudo llevar a cabo o no.
 - o `getFechaHora ()`. Devuelve la fecha y la hora de cada fichaje que existe en la base de datos, en forma de “`ArrayList`” de “`String`”.
 - o `listaFichajes ()`. Selecciona la información de todos los fichajes existentes en la base de datos. Devuelve un “`ArrayList`” de “`Fichaje`”.
 - o `verFichajes (String id)`. Obtiene la lista de fichajes de un profesor determinado que se elige desde el parámetro “`id`”. Devuelve un “`ArrayList`” de “`Fichaje`”.
 - o `justificarFalta (String texto, int id)`. Por medio de este método un profesor puede justificar sus faltas de asistencia, actualizando el campo de “excusa” de la tabla de fichajes. Se completa dicho campo con el “`texto`” que ha establecido el profesor en el fichaje indicado (`id`). Devuelve un booleano dependiendo de si la operación se pudo llevar a cabo o no.

- `getIdFichaje()` . Retorna el identificador (de tipo “int”) del último fichaje incorporado a la base de datos.
- `borrarFichaje (int id)` . Elimina el fichaje especificado por el identificador que se pasa como parámetro. Devuelve un booleano dependiendo de si la operación se pudo llevar a cabo o no.
- `setIdentry (int identry, int id)` . Establece el identificador de la reserva a la que se refiere el fichaje que se indica a través del parámetro “*id*”, en el caso de encontrarnos en un fichaje a clase. Devuelve un booleano dependiendo de si la operación se pudo llevar a cabo o no.
- `setTutoria (int tutoria, int id)` . Establece el identificador de la tutoría a la que se refiere el fichaje que se indica a través del parámetro “*id*”, en el caso de encontrarnos en un fichaje a tutoría. Devuelve un booleano dependiendo de si la operación se pudo llevar a cabo o no.
- `fichajesSinExcusas (String id)` . Devuelve la lista de fichajes que están sin justificar por el usuario cuyo identificador es “*id*”.
- `getInfoFichaje (int id)` . Con este método obtenemos la información completa de un fichaje determinado (“*id*”). Es utilizado cuando un profesor ficha correctamente y sirve para mostrar los datos del fichaje.

```

public List<Fichaje> listaFichajes() throws SQLException {
    ResultSet resultados = null;
    List<Fichaje> lista = null;
    try {
        Statement sentencia = ConexionUtil.openStatement();
        synchronized (sentencia) {
            resultados = sentencia
                .executeQuery("select mrbs_fichaje.id, email, fecha, identry, excusa, confianza, "
                    + "entrada_salida, tutoria from mrbs_fichaje "
                    + "inner join mrbs_usuario on mrbs_fichaje.id_usuario=mrbs_usuario.id");
        }
        if (!resultados.next()) {
            return null;
        } else {
            lista = new ArrayList<Fichaje>();
            resultados.beforeFirst();
            while (resultados.next()) {
                lista.add(new Fichaje(Integer.parseInt(resultados.getString(1)), resultados.getString(2),
                    resultados.getString(3), Integer.parseInt(resultados.getString(4)),
                    resultados.getString(5), Integer.parseInt(resultados.getString(6)),
                    resultados.getString(7), Integer.parseInt(resultados.getString(8))));
            }
        }
    } catch (SQLException e2) {
        throw e2;
    } finally {
        if (resultados != null) {
            try {
                resultados.close();
            } catch (SQLException ex) {
                Logger.getLogger(ProfesorController.class.getName()).log(Level.SEVERE, "No se pudo cerrar el Resulset", ex);
            }
        }
    }
    return lista;
}

```

Algoritmo 9: ejemplo de código en el *FichajeService* (método para mostrar la lista de fichajes completa)

- **InformeService**. Contiene los métodos relacionados con los informes que pueden generar los administradores. Solo dispone de dos métodos que sirven para cargar de datos los menús desplegables en dos de los cuatro informes:
 - `getAulas()` . Selecciona todas las aulas que existen en la base de datos para que el administrador pueda elegir el aula sobre la que quiere generar el informe de: “aula por día”. Devuelve una lista en forma de “`ArrayList`” del tipo “`Aula_Titulacion`”.
 - `getTitulaciones()` . Selecciona todas las titulaciones que existen en la base de datos para que el administrador pueda elegir la titulación sobre la que quiere generar el informe de: “titulación por día”. Devuelve una lista en forma de “`ArrayList`” del tipo “`Aula_Titulacion`”.

```

public List<Aula_Titulacion> getTitulaciones() throws SQLException {
    ResultSet resultados = null;
    List<Aula_Titulacion> titulaciones = null;
    try {
        Statement sentencia = ConexionUtil.openConnection();
        synchronized (sentencia) {
            resultados = sentencia
                .executeQuery("select * from mrbs_titulacion");
        }
        if (!resultados.next()) {
            return null;
        } else {
            titulaciones = new ArrayList<Aula_Titulacion>();
            resultados.beforeFirst();
            while (resultados.next()) {
                titulaciones.add(new Aula_Titulacion(resultados.getString(1), resultados.getString(2)));
            }
        }
    } catch (SQLException e2) {
        throw e2;
    } finally {
        // Se cierra resultSet
        if (resultados != null) {
            try {
                resultados.close();
            } catch (SQLException ex) {
                Logger.getLogger(ProfesorController.class.getName()).log(
                    Level.SEVERE, "No se pudo cerrar el Resulset", ex);
            }
        }
    }
    return titulaciones;
}
}

```

Algoritmo 10: ejemplo de código en el InformeService (método para obtener las titulaciones)

- **ReservaService**. Contiene los métodos relacionados con las reservas de aulas.
 - `buscarReserva(List<String> fechaHora, String entrada_Salida)` . Devuelve el identificador de la reserva de aula que se quiere buscar. Es decir, este método sirve para identificar una reserva a partir de la fecha y hora (`List<String> fechaHora`) en el que se realiza un fichaje de clase.

- o `verReservas (String user)`. Obtiene la lista de reservas de aulas (retorna un “`ArrayList`” del tipo “`Reserva`”) de un determinado profesor, partiendo del nombre de usuario (“`user`”) que se indica en el campo “`create_by`” de la tabla de reservas, que se corresponderá con el correo electrónico.

```

public int buscarReserva(List<String> fechaHora, String entrada_Salida) {
    int IDReserva=0;
    ResultSet resultados = null;
    String queryEntrada = "select id from mrbs_entry where date(from_unixtime(start_time))="
        + "str_to_date('" + fechaHora.get(0) + "', '%Y-%m-%d') and '" + fechaHora.get(1) +
        "' between subtime(time(from_unixtime(start_time)), '00:15:00') and "
        + "addtime(time(from_unixtime(start_time)), '00:15:00');";
    String querySalida = "select id from mrbs_entry where date(from_unixtime(start_time))="
        + "str_to_date('" + fechaHora.get(0) + "', '%Y-%m-%d') and '" + fechaHora.get(1) +
        "' between subtime(time(from_unixtime(end_time)), '00:15:00') and "
        + "addtime(time(from_unixtime(end_time)), '00:15:00');";
    try {
        Statement sentencia = ConexionUtil.openConnection();
        synchronized (sentencia) {
            if(entrada_Salida.equals("entrada")){
                resultados = sentencia
                    .executeQuery(queryEntrada);
            }else{
                resultados = sentencia
                    .executeQuery(querySalida);
            }
        }
        if (resultados.next() == false) {
            System.out.println("No hay resultados next");
            return -1;
        } else {
            IDReserva=Integer.parseInt(resultados.getString(1));
            System.out.println("ID: "+IDReserva);
            return IDReserva;
        }
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        if (resultados != null) {
            try {
                resultados.close();
            } catch (SQLException ex) {
                Logger.getLogger(ClaveServiceBD.class.getName()).log(
                    Level.SEVERE, "No se pudo cerrar el Resulset", ex);
            }
        }
    }
    return IDReserva;
}

```

Algoritmo 11: ejemplo de código en el `ReservaService` (método para buscar una reserva)

- **TutoriaService**. Contiene los métodos relacionados con las tutorías de un profesor.
 - o `buscarTutoria (List<String>fechaHora, String entrada_Salida)`. Devuelve el identificador de la tutoría que se quiere buscar. Es decir, este método sirve para identificar una tutoría a partir de la fecha y hora (`List<String> fechaHora`) en el que se realiza un fichaje de tutorías.
 - o `verTutorias (String id)`. Obtiene la lista de tutorías (retorna un “`ArrayList`” del tipo “`Tutoria`”) de un determinado profesor, partiendo del identificador del usuario (“`id`”).

```

public int buscarTutoria(List<String> fechaHora, String entrada_Salida) {
    int IDTutoria=0;
    ResultSet resultados = null;
    String queryEntrada = "select mrbs_tutoria.id from mrbs_tutoria inner join mrbs_periodo on "
        + "mrbs_tutoria.periodo=mrbs_periodo.id where dia=(weekday(now())+1) and '"+"
        + fechaHora.get(1)+"' between subtime(h_inicio, '00:30:00') and addtime(h_inicio, '00:30:00') and '"+"
        + fechaHora.get(0)+"' between mrbs_periodo.f_inicio and mrbs_periodo.f_fin";

    String querySalida = "select mrbs_tutoria.id from mrbs_tutoria inner join mrbs_periodo on "
        + "mrbs_tutoria.periodo=mrbs_periodo.id where dia=(weekday(now())+1) and '"+"
        + fechaHora.get(1)+"' between subtime(h_fin, '00:30:00') and addtime(h_fin, '00:30:00') and '"+"
        + fechaHora.get(0)+"' between DATE(from_unixtime(mrbs_periodo.f_inicio)) and "
        + "DATE(from_unixtime(mrbs_periodo.f_fin))";

    try {
        Statement sentencia = ConexionUtil.openStatement();
        synchronized (sentencia) {
            if(entrada_Salida.equals("entrada")){
                resultados = sentencia
                    .executeQuery(queryEntrada);
            }else{
                resultados = sentencia
                    .executeQuery(querySalida);
            }
        }
        if (resultados.next() == false) {
            System.out.println("No hay resultados next");
            return -1;
        } else {
            IDTutoria = Integer.parseInt(resultados.getString(1));
            System.out.println("IDTutoria"+IDTutoria);
            return IDTutoria;
        }
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        if (resultados != null) {
            try {
                resultados.close();
            } catch (SQLException ex) {
                Logger.getLogger(ClaveServiceBD.class.getName()).log(
                    Level.SEVERE, "No se pudo cerrar el Resulset", ex);
            }
        }
    }
}

```

Algoritmo 12: ejemplo de código en el TutoriaService (método para buscar una tutoría)

- **UsuarioService**. Contiene los métodos relacionados con los usuarios.
 - login(String email, String password). Verifica la identidad del usuario que trata de acceder con sus credenciales de correo (parámetro “email”) y contraseña (parámetro “password”). Comprueba que el correo y la contraseña coinciden, y, si es así, devuelve un verdadero de tipo booleano, en caso contrario, falso.
 - getUsuario (int userId). Devuelve toda la información del usuario buscado (objeto de tipo “Usuario”), que se indica a partir del identificador de dicho usuario (*userId*).
 - registro (Usuario user). Registra a un usuario con toda la información que indica en el formulario de registro, y lo inserta en la tabla que se corresponde: **mrbs_usuario** y **mrbs_users**, para que también pueda tener acceso a la aplicación de MRBS. Devuelve un booleano dependiendo de si la operación se pudo llevar a cabo o no.
 - getIdUsuario (Usuario user). Retorna el identificador del usuario que se busca a partir del correo electrónico (se obtiene del objeto “user” que se pasa como parámetro).

- `getUserByEmail (String email)`. Realiza lo mismo que el método anterior, de forma directa a partir del correo electrónico (parámetro “email”), pero en este caso se devuelve la información completa del usuario buscado (objeto de tipo “Usuario”).
- `getProfesores ()`. Útil para los informes. Selecciona todos los profesores que existen en la base de datos para que el administrador pueda elegir el profesor sobre la que quiere generar el informe de: “tutorías por profesor”. Devuelve una lista en forma de “ArrayList” del tipo “Usuario”.

```

public int getIdUsuario(Usuario user) {
    ResultSet resultados = null;
    int id=-1;
    try {
        Statement sentencia= ConexionUtil.openStatement();
        synchronized (sentencia) {
            resultados = sentencia
                .executeQuery("SELECT id FROM mrbs_usuario where email='"
                    + user.getEmail() + "'");
        }
        if (resultados.next()==false) {
            return -1;
        } else {
            id=Integer.parseInt(resultados.getString("id"));
            return id;
        }
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        // Se cierra resultSet
        if (resultados != null) {
            try {
                resultados.close();
            } catch (SQLException ex) {
                Logger.getLogger(UsuarioServiceBD.class.getName()).log(
                    Level.SEVERE, "No se pudo cerrar el Resulset", ex);
            }
        }
    }
    return id;
}

```

Algoritmo 13: ejemplo de código en el UsuarioService (método para obtener el identificador de un usuario)

6.1.3. Capa del modelo

La información que se obtiene de la base de datos se ve representada a través de varias clases que sirven para posteriormente ser tratadas en la capa de presentación. Se ha llevado a cabo los modelados de los datos siguientes:

- `Aula_Titulacion.java`. Representa la información a la vez de un aula o una titulación, de la que solo se hace constancia: nombre, id.

- Clave.java. Representa la información de una clave, contando con dos constructores diferentes (uno incorpora el identificador del usuario y otro el correo de usuario): id, usuario o email, huella.
- Fichaje.java. Representa la información más completa sobre un fichaje: id, idUsuario, fecha, identity, excusa, confianza, entrada_salida, start_time, end_time, room, name, description, tutoria. Se compone de diferentes constructores, dependiendo del tipo de fichaje (aula o tutoría), y de si se trata de representar únicamente los datos de la tabla de fichajes o no.
- Reserva.java. Representa la información acerca de una reserva de aula: start_time, end_time, nombre, usuario, room, tipoReserva, descripcion, status.
- Tutoria.java. Representa la información más completa acerca de una tutoría: dia, horalnicio, horaFin, despacho, periodo, periodolnicio, periodoFin, usuario, fecha.
- Usuario.java. Representa la información acerca de un usuario: id, nombre, apellidos, email, departamento, area, dedicacion, tipo, telefono, despacho, web_Personal, web_Investigacion, password.

Al margen de estas clases que simplemente se utilizan para obtener los datos que se recuperan de la base de datos, tenemos dos clases más que si tienen funcionalidades propias, y son las siguientes:

- Arduino.java. Se encarga de realizar la conexión de lectura y escritura con el Arduino y Java a través de su puerto serie. Envía datos al Arduino: petición de registrar una huella (“registro”), realizar un fichaje comprobando la identidad de la huella del profesor (“match”, y devuelve el identificador de la huella encontrada) y envío del identificador de una huella que se quiere eliminar. Estos métodos utilizan una clase llamada ConexionArduino.java, en la cual se tiene una librería (**PanamaHitek_Arduino**) con todas las funciones necesarias para conectar Arduino y Java. Ésta se encontrará en el paquete “util” donde también se sitúa ConexionUtil.java, la clase que permite la conexión con la base de datos a través del conector JDBC.
- Puerto_LecturaEscritura.java. Esta clase se ha desarrollado para permitir la funcionalidad de cambiar el nombre del puerto de conexión del Arduino. Almacena en un fichero de texto dicho nombre, y también existe otro método que permite leer el fichero.

Para poder realizar las modificaciones explicadas sobre la base de datos de MRBS combinada con la de Milagros y Jorge, se ejecutaron las siguientes sentencias SQL:

```
alter table mrbs_fichaje add column confianza int(3) not null;
alter table mrbs_fichaje add column entrada_salida varchar(7) not null;
alter table mrbs_fichaje add column tutoria int(11) not null;
```

Algoritmo 14: código para modificar la tabla mrbs_fichaje

```

CREATE TABLE mrbs_claves (
    id int(11) NOT NULL AUTO_INCREMENT,
    id_usuario int(11) NOT NULL,
    id_huella varbinary(50) NOT NULL,
    PRIMARY KEY (id),
    UNIQUE KEY id_huella (id_huella) ENGINE=InnoDB DEFAULT CHARSET=utf8
    COLLATE=utf8_bin AUTO_INCREMENT=1;

```

Algoritmo 15: código para crear la tabla mrbs_claves

```

CREATE TABLE mrbs_titulacion (
    id int(11) AUTO_INCREMENT NOT NULL,
    nombre varchar(36) NULL,
    PRIMARY KEY(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
    COLLATE=utf8_bin AUTO_INCREMENT=1;

```

Algoritmo 16: código para crear la tabla mrbs_titulacion

```

alter table mrbs_asigprof add column id_titulacion int (11) not null;
SET foreign_key_checks = 0;
alter table mrbs_asigprof add foreign key(id_titulacion) references mrbs_titulacion(id);
SET foreign_key_checks = 1;

```

Algoritmo 17: código para modificar la tabla mrbs_asigprof

6.2. Arquitectura Hardware

Pasemos ahora a la parte hardware de nuestra aplicación. Como ya indicamos en el anterior apartado de diseño, tendremos un **Arduino** al que se conecta un **lector de huellas** y una **pantalla LCD**, además de un servidor en una **Raspberry Pi 2**.

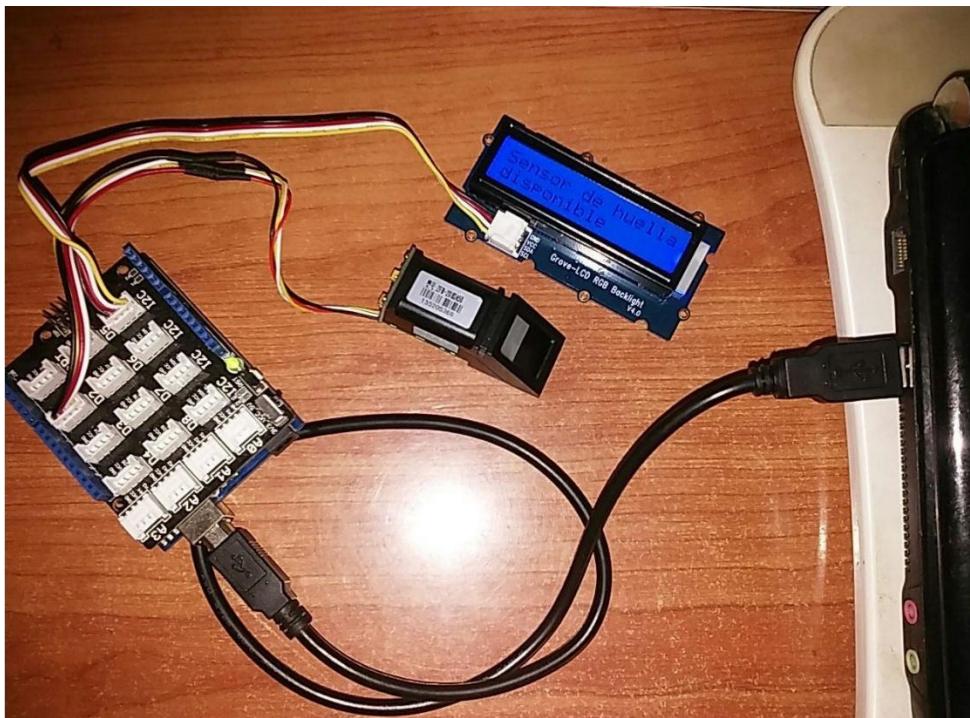


Figura 47: imagen de nuestro sistema de conexión Arduino

Para nuestro sistema, nos basta con utilizar **Arduino UNO** [71], modelo básico de estas placas microcontroladoras a la que conectaremos una **base shield de Grove** [72] que nos facilitará el trabajo en cuanto a conexiones y circuitos. Con ella, todos los puertos de entrada/salida del Arduino son sustituidos por conectores Grove.

Por lo tanto, tan sólo habrá que conectar los pines del sensor de huellas **Adafruit** a la placa Arduino, y hacerlo funcionar mediante el IDE, a través de la librería que proporciona Adafruit para permitir el completo uso de todas las funcionalidades que ofrece. Más adelante explicaremos el programa desarrollado para este proyecto [73].

Como ya vimos en el apartado 2.4.8, este lector es de tipo óptico y nos permitirá almacenar hasta 162 huellas diferentes por lo general, realizando verificación e identificación 1:1 y 1:N como cualquier sensor de huellas, y podrá llegar a tener cinco niveles de seguridad. Almacena las imágenes de las huellas como unas **templates** o **plantillas** en una memoria Flash interna del lector, y éstas son asociadas con un identificador único, los cuales guardamos en nuestra base de datos, en la tabla **mrbs_claves**.

Al igual que el lector Adafruit, conectaremos en otro puerto de la base shield de Grove una **pantalla LCD** la cual va mostrando las instrucciones para que un usuario utilice correctamente el sensor, además de informarlo en cuanto al resultado obtenido en el registro de la huella, en el fichaje o al realizar una eliminación de una huella. Esta pantalla irá cambiando de color dependiendo si está dando una **instrucción (azul)**, si se ha producido un **error (rojo)** o si se ha llevado a cabo con **éxito** la operación solicitada (**verde**).

En segundo lugar, en una **Raspberry Pi 2** instalaremos **MySQL** para desplegar nuestra base de datos MRBS y de fichajes, y tendremos el proyecto de Jorge Alvarado (aplicación MRBS conjunta a la gestión de las tutorías del profesorado) alojado en el servidor **Apache** que también será instalado en la Raspberry Pi.

Por otro lado, la página web de fichajes deberá estar almacenada en el sistema local donde se vaya a conectar el sensor, en un servidor **Apache Tomcat**.

Para completar la instalación de toda esta arquitectura hardware, véase los **anexos** que se adjuntan al final de esta memoria.

6.2.1. Programa Arduino

Por medio del entorno de desarrollo de Arduino, bajo el lenguaje C, se ha desarrollado una aplicación para que escuche varias acciones, llegadas desde nuestro programa Java.

Un algoritmo escrito en la plataforma Arduino se compone de dos funciones principales: **setup** y **loop** [67].

El **setup** es la primera función en ejecutarse dentro del programa, y es donde se inicializan las variables, se comienzan a usar las bibliotecas y/o funciones, se configuran el modo de trabajo de los pines (*INPUT* u *OUTPUT*), etc. Esta función es llamada cuando se inicia el Arduino, y sólo se ejecuta una vez [68].

En cuanto a la función **loop**, ésta es un bucle, es decir, se ejecuta un número infinito de veces tras haberse ejecutado el setup. Se detiene cuando se apaga o se reinicia el Arduino [69].

De esta forma, en nuestra aplicación, el `setup()` contiene lo siguiente:

- Antes del `setup` se inicializan las variables globales que se usa en el algoritmo, mientras que dentro, el `setup` se centra en arrancar el lector de huellas y la pantalla LCD, además de comprobar si el sensor está disponible y conectado o no. También establece los **baudios** (unidad de medida que representa el número de símbolos por segundos que transmite el Arduino y el lector), siempre ambos a **9600**.

En el bucle `loop()`, se encuentra un condicionante para determinar y llamar a la función que quiere realizar el programa Java, leyendo los datos que se han transmitido al Arduino:

- Si se recibe la palabra “**registro**”, se obtiene un identificador de huella libre con la función `obtenerID()`, y se registra la huella en el lector, por medio del método propio de la librería Adafruit llamado `getFingerprintEnroll()`. Se envía a la aplicación Java el identificador registrado.
- Si se recibe la palabra “**match**”, quiere decir que se pretende verificar la identidad del usuario antes de realizar un fichaje, y se comprueba por medio del método propio de la librería Adafruit llamado `getFingerprintID()`. Se envía a la aplicación Java el identificador de la huella encontrada, y la confianza con la que se ha verificado la identidad, básicamente, el nivel de igualdad entre las dos huellas que el usuario ha tenido que dejar para identificarse.
- En el último caso, se ejecutará la funcionalidad que queda, borrar una huella. Desde Java llega el identificador de la huella a eliminar, y por medio del método propio de la librería Adafruit llamado `deleteFingerprint()`, el sensor vacía el identificador y elimina la huella que tenía guardada. Se envía a la aplicación Java un **1** en señal de que se ha efectuado el borrado, y un **-1** si falló la operación.

```

int getFingerprintID() {
    lcd.clear();
    lcd.setRGB(0, 0, 255);
    lcd.print("Apoye su dedo"); lcd.setCursor(0, 1); lcd.print("sobre el sensor");
    uint8_t p = finger.getImage();
    if (p != FINGERPRINT_OK) return -1;

    p = finger.image2Tz();
    if (p != FINGERPRINT_OK) return -1;

    p = finger.fingerFastSearch();
    if (p != FINGERPRINT_OK) return -1;

    lcd.clear();
    lcd.setRGB(0, 255, 0);
    lcd.print("Encontrado: #");
    id=(finger.fingerID*1000)+finger.confidence;
    lcd.print(finger.fingerID);
    lcd.setCursor(0, 1);
    lcd.print("Confianza: ");
    lcd.print(finger.confidence);
    Serial.println(id);
    delay(2000);
    lcd.clear();
    return finger.fingerID;
}

```

Algoritmo 18: ejemplo de código en el programa Arduino (método para obtener una coincidencia de la huella)

6.2.2. Servidor Raspberry Pi 2

Para conocer la configuración e instalación del servidor en la placa Raspberry Pi 2, véase los anexos 1 y 2, en los cuáles se explica el proceso para instalar el servidor Apache y el gestor de base de datos MySQL, así como la integración de la aplicación MRBS.



Figura 48: imagen de nuestra Raspberry Pi 2 conectada a Internet por cable RJ-45, a la corriente a través de un Micro-USB, y a una pantalla por medio del HDMI. Cuenta con una tarjeta SD de 32 GB

6.3. Implementación de la seguridad

El último apartado de la implementación es la seguridad, los diferentes métodos que se han llevado a cabo para proteger cualquier dato que esté almacenado y sea sensible, además de que la aplicación desarrollada sea segura y fiable para lo que se va a utilizar en gran medida, los fichajes del profesorado.

6.3.1. Implementación de seguridad en la identificación biométrica

Como ya dijimos en el diseño de la seguridad, la identificación del profesor que pretende realizar un fichaje a sus clases y tutorías se asegura gracias al sistema biométrico que usamos para identificar al usuario, la **huella dactilar**.

Para no repetir lo dicho en el apartado 5.5.1, recordamos que el sensor que utilizamos de Adafruit implementa ya de manera interna ciertos niveles de seguridad, que por lo normal se sitúa en el **nivel 3**: tasa de falso rechazo menor al 1% y tasa de falsa aceptación menor de 0.001%.

6.3.2. Implementación de seguridad en el servidor y en la aplicación web

En el diseño de este apartado se abordaron las principales características que cumple la base de datos de nuestra aplicación.

Para la **Autentificación**, en la web, si no estás *logueado* la página te redirigirá al inicio, impidiendo así que puedan entrar intrusos sin acceder por usuario (correo electrónico) y contraseña. La aplicación MRBS tiene su propia autentificación, al igual que el gestor de tutorías, también por usuario y contraseña.

Además, para acceder a la base de datos es necesario autenticarse a través del único usuario de MySQL que existe con todos los privilegios sobre ella.

Como ya se ha dicho en la sección 6.1.1, tenemos dos tipos de usuarios diferentes: **administrador**, que puede crear los usuarios y alterar cierta información de la base de datos, y **profesor**, quien solo puede realizar consultas acerca de sus datos, tutorías... De esta forma se solventa el problema de **Autorización**.

Por parte de la **Privacidad**, para proteger la información sensible, se ha llevado a cabo una **encriptación AES**. Las contraseñas de los usuarios y los identificadores de las huellas almacenadas se cifran bajo este algoritmo, y se realiza simplemente con dos funciones:

- Insertar un nuevo dato:

```
INSERT INTO mrbs_usuario (password) VALUES (AES_ENCRYPT  
('password', 'clave'));
```

- Seleccionar datos encriptados:

```
SELECT AES_DECRYPT (id_huella, 'clave') from  
mrbs_claves;
```

Capítulo 7: Conclusiones y trabajos futuros

Contenidos

Capítulo 7: Conclusiones y trabajos futuros.....	91
7.1. Conclusiones.....	91
7.2. Trabajos futuros	92
7.3. Planificación estimada y planificación real	94

Para terminar, haremos un resumen de las conclusiones que se pueden extraer de este Trabajo Fin de Grado, además de los trabajos futuros que podrían realizarse y quedar pendientes tras este proyecto.

7.1. Conclusiones

La inmensa mayoría de las empresas y entornos de trabajo tienen la necesidad de **controlar la asistencia** de sus trabajadores, y, con la renovación que ha traído el **Plan Bolonia** en las universidades, esto también se hace necesario en el ámbito del **profesorado**.

Vivimos en una época en la que la evolución tecnológica nos invade y cobra cada vez más importancia en nuestros hábitos de vida. Por ello, cualquier sistema que pueda ser automatizado y digitalizado resultará al fin y al cabo más cómodo y productivo para todos.

Más concretamente, en el **Centro Universitario de Mérida** el control de la asistencia de los profesores se viene realizando a través de hojas de papel que incorporan unas tablas en las que el profesor firma manualmente sus horas. A este sistema surgió la alternativa del Trabajo Fin de Grado de Milagros Membrillo en el cual los fichajes se llevaban a cabo ya incorporando la tecnología, con una aplicación móvil que identificaba a los usuarios a través de sus tarjetas NFC.

Sin embargo, este sistema, como se dijo en los antecedentes de la introducción de esta memoria (1.2.), también presenta múltiples problemas e inconvenientes. Así, este Trabajo Fin de Grado ha realizado un estudio exhaustivo de los mejores métodos para identificar a una persona de manera única, segura e inequívoca, lo que llevó a analizar los diferentes **sistemas biométricos** que existen en la actualidad.

Por comodidad y sencillez nos decantamos por los **lectores de huellas**, una forma con la que los profesores no tengan que cargar con objetos para su identificación (además, estos pueden perderse y/o dejarse a otra persona para que esta fiche por él, perdiendo seguridad).

Mediante el **sensor Adafruit** compatible con **Arduino** se obtuvo una alternativa económica a la biometría de código cerrado. Este se conecta a un ordenador y desde la aplicación web desarrollada para este Trabajo Fin de Grado se interactúa con el lector de huellas, así como permitir visualizar los horarios, comprobar los fichajes realizados, ver las tutorías y clases de cada profesor... y todo desde un sistema web sencillo, simple y usable para cualquier tipo de usuario. La información de los profesores, horarios y fichajes se encuentra registrada de manera íntegra en una **Raspberry Pi 2** transformada en un servidor de base de datos.

En definitiva, las funcionalidades que se ofrecen en la aplicación para cada tipo de usuario son las siguientes:

- **Usuario administrador.** Son los únicos que pueden dar de alta a los profesores, así como configurar la conexión Arduino y eliminar huellas registradas. También pueden visualizar todos los fichajes que existan de cada uno de los profesores y generar distintos tipos de informes en PDF.
- **Usuario profesor.** Los profesores desde la principal interfaz pueden registrar su huella y realizar fichajes, además de acceder a la aplicación MRBS donde tienen la posibilidad de gestionar las reservas de aulas y las tutorías. Una vez se identifiquen en la web, el profesor incluso podrá ver su lista de fichajes, reservas y tutorías, así como justificar faltas de asistencia.

Esta breve conclusión demuestra que los objetivos fijados en el primer capítulo quedan abarcados y cubiertos: estudiar y analizar los sistemas hardware de lectura de huella dactilar y los diferentes SDK de código abierto o cerrado para desarrollar aplicaciones basadas en sensores de huellas, y diseñar y desarrollar una aplicación web desde la cual se controle el sistema de fichaje de entrada y salida a clase y a tutorías del profesor.

7.2. Trabajos futuros

Partiendo de este Trabajo Fin de Grado desarrollado, pueden surgir múltiples trabajos futuros, y de los posibles, los más importantes pueden ser:

7.2.1. Contenido Hardware

Con respecto a los dispositivos hardware que se han utilizado, siempre se puede **mejorar el lector de huellas** seleccionado si se decide invertir en sensores que no sean de código abierto.

En el caso de permanecer con el lector Adafruit, la conexión Arduino – Java podría mejorarse y sustituirse por otra forma por la que se pueda enviar y recibir información entre la aplicación y el sistema de Arduino, como puede ser que **el Arduino se prepare como un servidor que tan solo intercambie datos**. De esta forma, no se haría necesario tener físicamente un sensor junto al ordenador que gestione los fichajes, y así poder tener varios sensores distribuidos por las clases para que el profesor pueda desde un dispositivo móvil (la aplicación está orientada al diseño *responsive*) o desde su propio ordenador interactuar con el lector desde el aula donde tiene realizada la reserva, **saltando el paso de acudir a conserjería**. Esto también permitiría tener la web alojada en un servidor externo al ordenador local.

Incluso se podría **cambiar el sistema de biometría** por uno que aporte mucha más fiabilidad y seguridad, como pueden ser la verificación de patrones oculares, la geometría de la mano o las características faciales.

Un inconveniente que presenta el lector de Arduino es que tiene varias limitaciones. Las más importantes puede ser la pequeña **cantidad de huellas que puede almacenar**, y que no se pueden extraer las imágenes de las huellas para su plena manipulación, es decir, para que estas puedan ser almacenadas en nuestra propia base de datos.

Otro aspecto interesante puede ser el cambio de un lector de huellas óptico, como es Adafruit, a un **lector de capacitancia** que generan las imágenes con corriente eléctrica, y solventan el pequeño problema que puede existir con algunos de los sensores ópticos reflexivos de ser vulnerados mediante imágenes.

Para acabar con las mejoras en el hardware, sería recomendable **incrementar la seguridad del servidor**, así como sustituir la Raspberry Pi 2 por un servidor real.

7.2.2. Contenido Software

Por otro lado, siempre se puede incluir mejoras en la aplicación web desarrollada, ya sean de diseño, añadido de funcionalidades, actualización del código, etc.

En cuanto a la incorporación de nuevas funciones, estas podrían ser:

- Aumentar el **número de informes** diferentes que se pueden generar.
- **Muestra de los datos** de manera más selectiva. Por ejemplo, que un administrador pueda ver solo los fichajes de un único profesor o de varios, que es como está actualmente.
- Actualizar el sistema back-end de la web a las **tecnologías más recientes** (MyBatis, JPA, Spring MVC, Hibernate...). El front-end se encuentra con las últimas versiones de HTML y CSS, pero hay que estar atentos, pues con el tiempo pueden irse quedando obsoletas.
- Incluir el **control de la asistencia del alumnado** si hiciera necesario.
- Asegurar y **cifrar los datos** de la base de datos con mayor seguridad.

7.3. Planificación estimada y planificación real

La siguiente tabla muestra la planificación que finalmente se ha llevado a cabo de manera real mediante un **diagrama de Gantt**, y posteriormente se explican las diferencias que han ocurrido con respecto a la planificación inicial estimada (4.5):

	FEBRERO	MARZO	ABRIL	MAYO	JUNIO	JULIO	AGOSTO	SEPTIEMBRE	OCTUBRE	NOVIEMBRE
Estudiar y analizar los sistemas hardware de lectura de huella dactilar (40 horas)	40 horas									
Estudiar y analizar los distintos SDK libres y de pago para el desarrollo de aplicaciones basadas en la lectura de huella dactilar (20 horas)		20 horas								
Diseño y desarrollo de sistema de fichaje de entrada y salida a clase del profesor (315 horas)		15 horas	20 horas	20 horas	70 horas	70 horas	20 horas	60 horas	30 horas	10 horas
Redacción de memoria (75 horas)	10 horas	10 horas	5 horas			15 horas		20 horas		15 horas

Tabla 33: planificación real

En primer lugar, el análisis de los sistemas hardware de huella dactilar se demoró por el estudio más profundizado que se realizó previamente acerca de los diferentes sistemas biométricos. En cuanto al software necesario o SDK para estos sensores, la planificación no estuvo lejos de lo real.

En segundo lugar, la búsqueda de un lector de huellas económico y de código abierto, así como la decisión final de decantarnos por Adafruit, consumió demasiado tiempo y el primer objetivo de entregar este Trabajo Fin de Grado en junio o julio se hizo imposible.

Entre marzo y junio se instaló la configuración del servidor en la Raspberry Pi 2 y se estudió y desarrolló el programa de Arduino, algo que nos llevó más tiempo del esperado debido a la escasa y limitada documentación sobre el lector Adafruit. A la vez, se iba diseñando la aplicación web y se investigaba de cómo poder interactuar entre la aplicación y Arduino, para tener la posibilidad de enviar y recibir datos entre ambas partes. Se probó varias soluciones, pero ninguna satisfacía nuestra necesidad de intercambiarse datos entre una web y el Arduino, hasta que se encontró la solución de la librería PanamaHitek Java – Arduino.

Ya en verano se decidió llevar el trabajo más relajado y descansar. Por ello, en agosto solo se hicieron los informes con JasperReports y ya en septiembre se fue acabando el desarrollo web.

Por último, la memoria se fue madurando a lo largo de los meses, y, con la opción de entregar en septiembre descartada, se avanzó en gran medida en este mes con vistas ya a la presentación de noviembre. Por ello, en octubre y noviembre se fueron puliendo los últimos detalles y realizando las diferentes pruebas. La redacción de esta memoria se ha alargado pero muy poco, debido al estudio adicional de la biometría.

Capítulo 8: Bibliografía

- [1] *Estatutos de la Universidad de Extremadura*. (2010). Obtenido de
http://www.unex.es/organizacion/gobierno/sec_gral/archivos/ficheros/Normativas/DocumentointernodelosESTATUTOS.pdf
- [2] *Biometría*. (25 de Mayo de 2016). Obtenido de Wikipedia:
<https://es.wikipedia.org/wiki/Biometr%C3%A9tica>
- [3] Mifsud-k idatzia, E. (27 de Enero de 2012). *Sistemas físicos y biométricos de seguridad*. Obtenido de
<http://recursostic.educacion.es/observatorio/web/eu/cajon-de-sastre/38-cajon-de-sastre/1045-sistemas-fisicos-y-biometricos-de-seguridad>
- [4] Tolosa Borja, C., & Giz Bueno, Á. (s.f.). *Sistemas Biométricos*. Obtenido de
https://www.dsi.uclm.es/personal/MiguelFGraciani/mikicurri/Docencia/Bioinformatica/web_BIO/Documentacion/Trabajos/Biometria/Trabajo%20Biometria.pdf
- [5] Delgado, A. (6 de Febrero de 2012). *Seguridad biométrica en sistemas informáticos*. Obtenido de Eroski Consumer:
<http://www.consumer.es/web/es/tecnologia/software/2012/02/06/206350.php>
- [6] *Lectores de huella digital*. (s.f.). Obtenido de Biometricos.net:
<http://www.biometricos.net/search/label/Lectores%20de%20huella%20digital>
- [7] Sánchez Ávila, C. (2012). *Aplicaciones de la Biometría a la Seguridad*. Obtenido de Universidad Politécnica de Madrid - CEDINT:
http://www.criptored.upm.es/descarga/TASSI2012_CarmenSanchez.pdf
- [8] Cruceña, B. (13 de Abril de 2009). *Qué son los Sistemas Biométricos*. Obtenido de eju.tv.
- [9] *Biometría Informática*. (s.f.). Obtenido de UNAM - Facultad de Ingeniería:
<http://redyseguridad.fi-p.unam.mx/proyectos/biometria/index.html>
- [10] González Isabel, J. (4 de Febrero de 2013). *Sistema de identificación biométrica basado en huella dactilar mediante binarización sobre plataformas Android*. Obtenido de Universidad Carlos III de Madrid: http://e-archivo.uc3m.es/bitstream/handle/10016/19246/TFG_GONZALEZ_ISABEL_JOS_E_%20RAMON.pdf?sequence=1
- [11] *Tipos de Sistemas Biométricos*. (s.f.). Obtenido de
<http://seginfitzelestrada.blogspot.com.es/p/sistemas-biometricos.html>

- [12] *El olor corporal como identificador biométrico.* (s.f.). Obtenido de Campus Montegancedo - Universidad Politécnica de Madrid:
<http://www.upm.es/Montegancedo/Montegancedo/Noticias/fa1e38c6f97f3410VgnVCM10000009c7648aRCRD>
- [13] Pérez Díaz, A. J. (4 de Marzo de 2010). *Acceso a aplicaciones y al equipo con huella dactilar.* Obtenido de Proyecto AJPD Soft:
<http://ajpdsoft.com/modules.php?name=News&file=print&sid=451>
- [14] *Cómo funcionan los lectores de huella digital.* (2003). Obtenido de TEC Electrónica - MEX: <https://tec-mex.com.mx/promos/bit/bit0903-bio.htm>
- [15] Pérez, E. (20 de Octubre de 2013). *¿Cómo funcionan los lectores de huella dactilar?* Obtenido de Omicrono:
<http://www.omicrono.com/2013/10/borrador-lectores-de-huella-dactilares/>
- [16] *Tipos de lectores de huellas dactilares.* (s.f.). Obtenido de GuíasPrácticas.COM:
<http://www.guiaspracticas.com/controles-de-acceso/tipos-de-lectores-de-huellas-dactilares>
- [17] *Sensor de huella digital.* (19 de Noviembre de 2015). Obtenido de Wikipedia:
https://es.wikipedia.org/wiki/Sensor_de_huella_digital
- [18] Julián, G. (11 de Septiembre de 2013). *Tu huella es la entrada al smartphone: así funciona el sensor biométrico del nuevo iPhone 5S.* Obtenido de Xataka:
<http://www.xataka.com/moviles/tu-huella-es-la-entrada-al-smartphone-asi-funciona-el-sensor-biometrico-del-nuevo-iphone-5s>
- [19] *Fingerprint Readers.* (2014). Obtenido de Griaule Biometrics:
http://www.griaulebiometrics.com/en-us/fingerprint_sdk/supported_readers
- [20] *Lectores y escáneres de huellas biométrico.* (s.f.). Obtenido de Fulcrum Biometrics:
<http://es.fulcrumbiometrics.com/el-es-q80h/Dispositivos-biometricos-escaner-de-huellas-digitales-y-la-tecnologia-de-lector?searching=Y&sort=1&cat=34&show=100&page=1>
- [21] *Lector U.are.U 4500.* (2008). Obtenido de digitalPersona:
http://www.barmax.com/images/digitalpersona_barMax.pdf
- [22] *Lumidigm Venus Series sensors.* (2016). Obtenido de NEUROtechnology:
<http://www.neurotechnology.com/fingerprint-scanner-lumidigm-venus.html>
- [23] *NITGEN Fingkey Hamster.* (2016). Obtenido de NEUROtechnology:
<http://www.neurotechnology.com/fingerprint-scanner-nitgen-fingkey-hamster.html>
- [24] *Atmel Fingerchip sensors family.* (s.f.). Obtenido de NEUROtechnology:
<http://www.neurotechnology.com/fingerprint-scanner-atmel-fingerchip.html>

- [25] *FbF MobileOne*. (s.f.). Obtenido de Fulcrum Biometrics:
http://www.biometriaaplicada.com/E-STORE/DataSheets/FulcrumBio_FbFmobileOne_ds_email.pdf
- [26] *UPEK TouchChip TCRU1C*. (2016). Obtenido de NEUROtechnology:
<http://www.neurotechnology.com/fingerprint-scanner-ukek-touchchip-tcru1c.html>
- [27] *Fingerprint Biometric Scanner GT-511C3 / GT-511C31*. (2009). Obtenido de ADH Technology Co. Ltd.: [http://www.adh-tech.com.tw/?9,gt-511c3-gt-511c31-\(uart\)](http://www.adh-tech.com.tw/?9,gt-511c3-gt-511c31-(uart))
- [28] *Lector de huellas dactilares GT-511C3*. (s.f.). Obtenido de BricoGeek:
<http://tienda.bricogeek.com/sensores-imagen/630-lector-de-huellas-dactilares-gt-511c3.html>
- [29] HeTPro. (11 de Enero de 2016). *Lector de Huella Digital Arduino*. Obtenido de instructables.com: <http://www.instructables.com/id/Lector-De-Huella-Digital-Arduino/>
- [30] *Fingerprint Sensor Adafruit Arduino*. (s.f.). Obtenido de Adafruit:
<https://www.adafruit.com/products/751>
- [31] *Soluciones biométricas de huella digital*. (2012). Obtenido de Biometría Aplicada:
<http://www.biometriaaplicada.com/huella.html>
- [32] *Kits de desarrollo de software biométrico*. (s.f.). Obtenido de Fulcrum Biometrics:
<http://es.fulcrumbiometrics.com/el-es-q80q/Kit-de-desarrollo-de-Software-biometrico-biometrico-SDK?searching=Y&sort=1&cat=1&show=60&page=1>
- [33] *Librerías de desarrollo SDK para lectores ZK Software*. (2010). Obtenido de ACCESO: http://www.huellasdigitales.com.mx/sdk_zk.html
- [34] *DigitalPersona Pro Enterprise*. (2012). Obtenido de
[http://www.biometriaaplicada.com/E-STORE/DataSheets/ds-DigitalPersonaProEnterprise-20120302%20\(2\).pdf](http://www.biometriaaplicada.com/E-STORE/DataSheets/ds-DigitalPersonaProEnterprise-20120302%20(2).pdf)
- [35] *VeriFinger SDK*. (2016). Obtenido de NEUROtechnology:
http://www.neurotechnology.com/vf_sdk.html
- [36] *IDKit SDK*. (s.f.). Obtenido de Innovatrics:
<http://www.innovatrics.com/es/products/idkit>
- [37] *SDK Gratuito para verificación dactilar*. (s.f.). Obtenido de GOIT.cl:
<http://www.goit.cl/freesdk.html>
- [38] *Lector de huella digital Suprema BioMini*. (s.f.). Obtenido de Kimaldi Electronics:
http://www.kimaldi.com/productos/sistemas_biometricos/lectores_de_huella_digital_para_pc/lector_de_huella_digital_suprema_biomini/sdk_para_el_software_suprema_biomini

- [39] *Desarrollo en Cascada Vs. Desarrollo Iterativo e Incremental.* (s.f.). Obtenido de <http://isescom.blogspot.com.es/2013/08/desarrollo-en-cascada-vs-desarrollo.html>
- [40] Palom, S., Bernal, A., & Rodríguez, T. (s.f.). *Desarrollo iterativo e incremental.* Obtenido de <http://player.slideplayer.es/19/5944207/#>
- [41] *Metodología de desarrollo iterativo y creciente.* (s.f.). Obtenido de EcuRed: http://www.ecured.cu/Metodolog%C3%ADA_de_desarrollo_iterativo_y_creciente
- [42] *Modelos de desarrollo de software.* (19 de Agosto de 2013). Obtenido de El Conspirador: <http://www.elconspirador.com/2013/08/19/modelos-de-desarrollo-de-software/>
- [43] Berzal, F. (s.f.). *El ciclo de vida de un sistema de información.* Obtenido de <http://elvex.ugr.es/idbis/db/docs/lifecycle.pdf>
- [44] Pressman, R. (s.f.). *Ingeniería del Software.*
- [45] *Access Control Using Biometrics Features with Arduino Galileo.* (8 de Agosto de 2014). Obtenido de http://www.ijarcsse.com/docs/papers/Volume_4/8_August2014/V4I8-0301.pdf
- [46] Pavón Mestras, J. (s.f.). *El patrón Modelo-Vista-Controlador (MVC).* Obtenido de Universidad Complutense de Madrid: <https://www.fdi.ucm.es/profesor/jpavon/poo/2.14.MVC.pdf>
- [47] *La arquitectura MVC.* (s.f.). Obtenido de LibrosWeb: http://librosweb.es/libro/jobeet_1_4/capitulo_4/la_arquitectura_mvc.html
- [48] *¿Qué es Arduino?* (s.f.). Obtenido de <https://proyectoarduino.wordpress.com/%C2%BFque-es-arduino/>
- [49] *¿Qué es Arduino?* (s.f.). Obtenido de ARDUINO.cl: <http://arduino.cl/que-es-arduino/>
- [50] *Raspberry Pi.* (s.f.). Obtenido de <https://www.raspberrypi.org/help/what-is-a-raspberry-pi/>
- [51] *Análisis de Aplicación: MRBS.* (s.f.). Obtenido de <http://www.bilib.es/recursos/catalogo-de-aplicaciones/analisis/doc/analisis-de-aplicacion-mrbs/docctrl/show/Documento/>
- [52] *Fingerprint Identification Module: User's Manual.* (s.f.). Obtenido de <https://sicherheitskritisch.de/files/specifications-2.0-en.pdf>
- [53] Diseño de un sistema de seguridad basado en procesamiento de imágenes para el acceso vehicular a un campus. (2010). INGENIUM. Obtenido de <http://gdsproc.com/revista/2010%20vera.pdf>

- [54] *Integridad Referencial*. (s.f.). Obtenido de
https://es.wikipedia.org/wiki/Integridad_referencial
- [55] *Como encriptar contraseñas de forma segura en una Base de datos*. (9 de octubre de 2015). Obtenido de <http://www.pacomaldonado.com/como-encriptar-contrasenas-de-forma-segura-en-una-base-de-datos/>
- [56] *Almacenar contraseñas en MySQL*. (s.f.). Obtenido de
<http://www.solingest.com/blog/almacenar-contrasenas-en-mysql>
- [57] Vinda, E. (s.f.). *Algoritmo AES*. Obtenido de
<http://es.slideshare.net/elvisvinda/sencilla-explicacion-sobre-aes>
- [58] *Advanced Encryption Standard*. (s.f.). Obtenido de
https://es.wikipedia.org/wiki/Advanced_Encryption_Standard
- [59] *Hoja de estilos en cascada*. (s.f.). Obtenido de
https://es.wikipedia.org/wiki/Hoja_de_estilos_en_cascada
- [60] *JQuery*. (s.f.). Obtenido de <https://es.wikipedia.org/wiki/JQuery>
- [61] *JavaScript*. (s.f.). Obtenido de <https://es.wikipedia.org/wiki/JavaScript>
- [62] *Bootstrap*. (s.f.). Obtenido de <http://getbootstrap.com/>
- [63] *Jaspersoft*. (s.f.). Obtenido de <http://community.jaspersoft.com/>
- [64] *Apache Tomcat*. (s.f.). Obtenido de <http://tomcat.apache.org/>
- [65] *Eclipse*. (s.f.). Obtenido de <https://eclipse.org/>
- [66] *Patrón de arquitectura Modelo Vista Controlador (MVC)*. (s.f.). Obtenido de
<http://www.lab.inf.uc3m.es/~a0080802/RAI/mvc.html>
- [67] *El Setup y el Loop en Arduino*. (s.f.). Obtenido de <http://panamahitek.com/el-setup-y-el-loop-en-arduino/>
- [68] *Setup Arduino*. (s.f.). Obtenido de <https://www.arduino.cc/en/Reference/Setup>
- [69] *Loop Arduino*. (s.f.). Obtenido de <https://www.arduino.cc/en/Reference/Loop>
- [70] *Oracle Java JDBC*. (s.f.). Obtenido de
<http://www.oracle.com/technetwork/java/javase/jdbc/index.html>
- [71] *Arduino Uno*. (s.f.). Obtenido de
<https://www.arduino.cc/en/Main/ArduinoBoardUno>
- [72] *Grove - Base Shield*. (s.f.). Obtenido de http://www.seeedstudio.com/wiki/Grove_-_Base_Shield
- [73] *Lector De Huella Digital Arduino*. (s.f.). Obtenido de
<http://www.instructables.com/id/Lector-De-Huella-Digital-Arduino/>

- [74] Álvarez, M. (8 de julio de 2002). *Qué es JSP*. Obtenido de <http://www.desarrolloweb.com/articulos/831.php>
- [75] *HTML5*. (s.f.). Obtenido de <https://developer.mozilla.org/es/docs/HTML/HTML5>
- [76] *MRBS Authentication*. (s.f.). Obtenido de http://mrbs.sourceforge.net/view_text.php?section=Documentation&file=AUTENTICATION
- [77] *MRBS*. (s.f.). Obtenido de <http://mrbs.sourceforge.net/>
- [78] *Apache*. (s.f.). Obtenido de <https://httpd.apache.org/>
- [79] *PHP*. (s.f.). Obtenido de <http://php.net/>
- [80] *MySQL*. (s.f.). Obtenido de <http://dev.mysql.com/>
- [81] *PHPMyAdmin*. (s.f.). Obtenido de <https://www.phpmyadmin.net/>
- [82] *Tutorial Raspberry Pi - Instalación de APACHE + MYSQL + PHP*. (s.f.). Obtenido de <https://geekytheory.com/tutorial-raspberry-pi-15-instalacion-de-apache-mysql-php/>
- [83] *Instalación MRBS*. (s.f.). Obtenido de <https://historialdelbecario.wordpress.com/2014/07/02/instalacion-mrbs/>
- [84] *Poner la dirección IP fija en Raspbian*. (s.f.). Obtenido de <https://raspberryparatorpes.net/installacion/poner-la-direccion-ip-fija-en-raspbian/>
- [85] *NO - IP*. (s.f.). Obtenido de <https://my.noip.com>
- [86] *Arduino IDE Software*. (s.f.). Obtenido de <https://www.arduino.cc/en/Main/Software>
- [87] *Adafruit Fingerprint Sensor Library*. (s.f.). Obtenido de <https://github.com/adafruit/Adafruit-Fingerprint-Sensor-Library>
- [88] *SFG Demo*. (s.f.). Obtenido de <https://drive.google.com/file/d/0B-J8rOkni00pRIBwWnJCVm1IUXM/view>
- [89] *Grove - Fingerprint Sensor*. (s.f.). Obtenido de http://wiki.seeedstudio.com/wiki/Grove_-_Finger_Print_Sensor_V1.0#Resource
- [90] *Grove - LCD RGB Backlight*. (s.f.). Obtenido de http://www.seeedstudio.com/wiki/Grove_-_LCD_RGB_Backlight
- [91] *Librería PanamaHitek Arduino*. (s.f.). Obtenido de http://panamahitek.com/libreria-panamahitek_arduino/
- [92] *Libería RXTX*. (s.f.). Obtenido de <http://rxtx.qbang.org/pub/rxtx/rxtx-2.1-7-bins-r2.zip>

- [93] *JasperSoft Studio*. (s.f.). Obtenido de
<https://marketplace.eclipse.org/content/jaspersoft-studio>
- [94] *XAMPP*. (s.f.). Obtenido de <https://www.apachefriends.org/es/index.html>
- [95] *Descarga de XAMPP*. (s.f.). Obtenido de
<https://www.apachefriends.org/es/download.html>
- [96] *Java JDK*. (s.f.). Obtenido de
<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html?ssSourceSiteId=otnes>
- [97] *Requisitos para utilizar Arduino con Java* (s.f.). Obtenido de
<http://panamahitek.com/requisitos-para-utilizar-arduino-con-java/>

Capítulo 9: Anexos

Contenidos

Capítulo 9: Anexos.....	103
9.1. ANEXO I: Instalación de Apache, MySQL y PHP	104
9.1.1. Introducción	104
9.1.2. Apache	104
9.1.3. PHP.....	105
9.1.4. MYSQL y PHPMyAdmin	105
9.2. ANEXO II: Instalación de MRBS.....	108
9.2.1. Introducción	108
9.2.2. Añadir gestor de tutorías	109
9.2.3. Establecer una dirección IP fija	110
9.3. ANEXO III: Arduino y lector de huellas Adafruit	112
9.3.1. Introducción	112
9.3.2. Instalación de Arduino IDE	112
9.3.3. Instalación de la librería Adafruit	112
9.3.4. Conexiones entre el Arduino, el lector de huellas y la pantalla LCD ...	113
9.3.5. Conexión Java - Arduino.....	115
9.3.6. Puerto de conexión Arduino.....	115
9.4. ANEXO IV: Jasper Reports	117
9.4.1. Introducción	117
9.4.2. Instalación de Jasper Reports en Eclipse	117
9.4.3. Creación de informes con Jasper Reports	118
9.5. ANEXO V: Despliegue de la aplicación web.....	124
9.5.1. Introducción	124
9.5.2. Instalación de XAMPP	124

9.1. ANEXO I: Instalación de Apache, MySQL y PHP

9.1.1. Introducción

En primer lugar, vamos a proceder a explicar los pasos que hay que seguir para la instalación de **Apache**, **MySQL** y **PHP**, requisitos de MRBS que contamos en el apartado 5.4 de la aplicación **MRBS** en la parte servidora. En nuestro caso, desplegaremos el servidor sobre una **Raspberry Pi 2** [82].

9.1.2. Apache

Necesitaremos un servidor web sobre el que desplegaremos la aplicación web MRBS. Para ello, haremos uso de Apache, proyecto de **Apache Software Foundation** que ha desarrollado un servidor HTTP de código abierto para los nuevos sistemas operativos, incluyendo UNIX y Windows. Su objetivo es proporcionar un servidor seguro, eficiente y extensible que proporciona servicios HTTP en sincronización con los estándares HTTP actuales [78].

Los comandos a ejecutar sobre el terminal en modo superusuario de Linux para conseguir tener Apache en nuestro servidor serán los siguientes:

- **addgroup www-data**
- **usermod -a -G www-data www-data**
- **apt-get install apache2**
- **/etc/init.d/apache2 restart**
- **service apache2 status**

Con este último comando podemos comprobar el estado de Apache, es decir, si se ha instalado de manera correcta en nuestro sistema:

```

pi@raspberrypi: ~
Active: inactive (dead)
pi@raspberrypi:~ $ sudo service apache2 status
● apache2.service - LSB: Apache2 web server
  Loaded: loaded (/etc/init.d/apache2)
  Active: active (running) since Wed 2016-06-01 14:52:58 UTC; 1 day 19h ago
    Process: 6025 ExecReload=/etc/init.d/apache2 reload (code=exited, status=0/SUC
CESS)
    Process: 521 ExecStart=/etc/init.d/apache2 start (code=exited, status=0/SUCCE
S)
      CGroupl: /system.slice/apache2.service
      └─829 /usr/sbin/apache2 -k start
         ├─6074 /usr/sbin/apache2 -k start
         ├─6075 /usr/sbin/apache2 -k start
         ├─6076 /usr/sbin/apache2 -k start
         ├─6077 /usr/sbin/apache2 -k start
         └─6078 /usr/sbin/apache2 -k start

Jun 01 14:52:57 raspberrypi apache2[521]: Starting web server: apache2AH0055...
Jun 01 14:52:57 raspberrypi apache2[521]: AH00558: apache2: Could not reliab...e
Jun 01 14:52:58 raspberrypi apache2[521]: .
Jun 01 14:52:58 raspberrypi systemd[1]: Started LSB: Apache2 web server.
Jun 03 06:25:06 raspberrypi systemd[1]: Reloading LSB: Apache2 web server.
Jun 03 06:25:07 raspberrypi apache2[6025]: Reloading web server: apache2.
Jun 03 06:25:07 raspberrypi systemd[1]: Reloaded LSB: Apache2 web server.
Jun 03 10:36:01 raspberrypi systemd[1]: Started LSB: Apache2 web server.
Hint: Some lines were ellipsized, use -l to show in full.
pi@raspberrypi:~ $ [ ]

```

Figura 49: el comando “service apache2 status” nos muestra que Apache está activo

9.1.3. PHP

MRBS necesita tener soporte PHP en el servidor web. **PHP** es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML, con el que se ha desarrollado MRBS [79].

La instalación de PHP en su versión 5 es tan simple como ejecutar en modo superusuario el comando:

➤ **apt-get install php5 libapache2-mod-php5**

9.1.4. MySQL y PHPMyAdmin

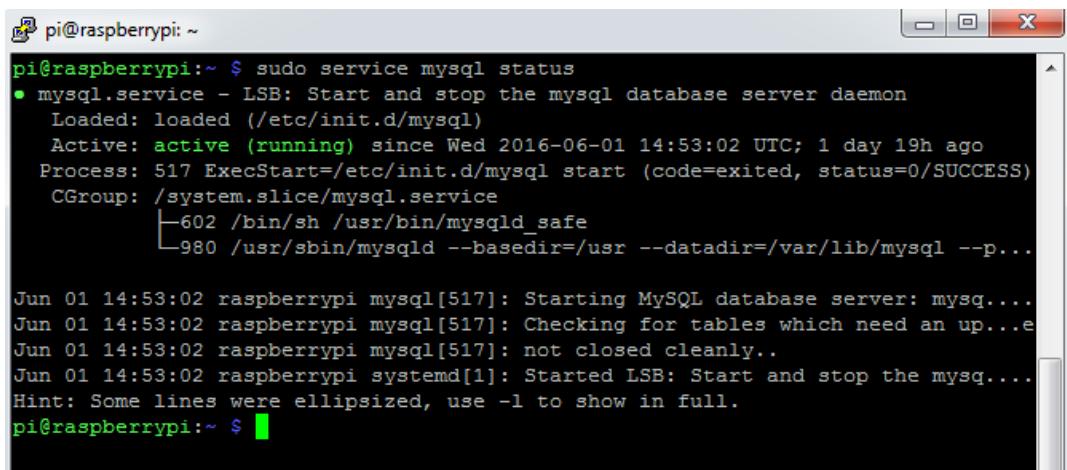
El último requisito de MRBS es un soporte de base de datos MySQL o PostgreSQL. Nos decantamos por **MySQL**, un sistema de gestión de base de datos relacional *open source* desarrollado bajo licencia GPL/licencia comercial por *Oracle Corporation*. Es la más popular para entornos de desarrollo web [80].

En la siguiente secuencia de comandos también instalamos **PHPMyAdmin**, una herramienta de software libre escrito en PHP que nos facilitará el manejo de la administración de MySQL [81]:

➤ **ifup lo**
 ➤ **apt-get install mysql-server mysql-client php5-mysql
phpmyadmin**

- contraseña para usuario root: **raspberry**
- contraseña para el usuario administrador: **raspberry**
- contraseña para la aplicación MySQL de PHPMyAdmin: **raspberry**

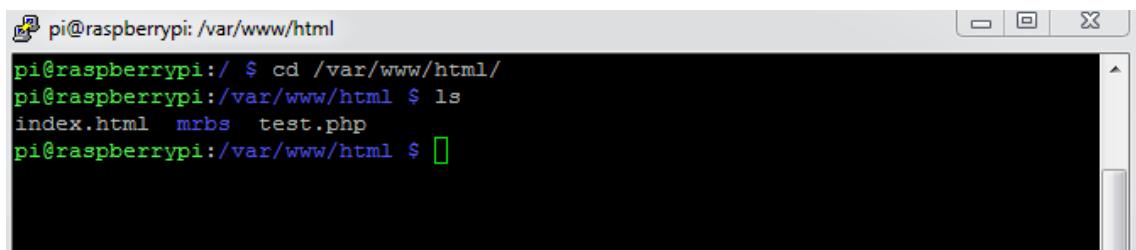
- **nano /etc/php5/apache2/php.ini.** Editamos este archivo añadiendo antes de la línea "Dynamics Extensions": **extension=mysql.so**.
- **ln -s /etc/phpmyadmin/apache.conf /etc/php5/apache2/conf.d/phpmyadmin.conf**
- **/etc/init.d/apache2 reload**
- **nano /etc/apache2/apache2.conf.** Al final de este archivo tenemos que agregar:
Include /etc/phpmyadmin/apache.conf
- **/etc/init.d/apache2 restart**
- Podremos comprobar el funcionamiento correcto de MySQL con **service mysql status**:



```
pi@raspberrypi:~ $ sudo service mysql status
● mysql.service - LSB: Start and stop the mysql database server daemon
  Loaded: loaded (/etc/init.d/mysql)
  Active: active (running) since Wed 2016-06-01 14:53:02 UTC; 1 day 19h ago
    Process: 517 ExecStart=/etc/init.d/mysql start (code=exited, status=0/SUCCESS)
   CGroup: /system.slice/mysql.service
           ├─602 /bin/sh /usr/bin/mysqld_safe
           └─980 /usr/sbin/mysqld --basedir=/usr --datadir=/var/lib/mysql --p...
Jun 01 14:53:02 raspberrypi mysql[517]: Starting MySQL database server: mysq...
Jun 01 14:53:02 raspberrypi mysql[517]: Checking for tables which need an up...
Jun 01 14:53:02 raspberrypi mysql[517]: not closed cleanly..
Jun 01 14:53:02 raspberrypi systemd[1]: Started LSB: Start and stop the mysq...
Hint: Some lines were ellipsized, use -l to show in full.
pi@raspberrypi:~ $
```

Figura 50: el comando “service mysql status” nos muestra que MySQL está activo

Si incorporamos en la carpeta **html** de **www**, en **var**, archivos web HTML o PHP, podremos acceder a ellos por medio de un navegador indicando la dirección IP del servidor y el nombre del fichero. Esto también ocurre con PHPMyAdmin.



```
pi@raspberrypi:/var/www/html
pi@raspberrypi:/ $ cd /var/www/html/
pi@raspberrypi:/var/www/html $ ls
index.html  mrbs  test.php
pi@raspberrypi:/var/www/html $
```

Figura 51: las aplicaciones web se despliegan en la carpeta html de www (Apache)

PHP Version 5.6.20-0+deb8u1		
System	Linux raspberrypi 4.1.13-v7+ #826 SMP PREEMPT Fri Nov 13 20:19:03 GMT 2015 armv7l	
Build Date	Apr 27 2016 23:08:42	
Server API	Apache 2.0 Handler	
Virtual Directory Support	disabled	
Configuration File (php.ini) Path	/etc/php5/apache2	
Loaded Configuration File	/etc/php5/apache2/php.ini	
Scan this dir for additional .ini files	/etc/php5/apache2/conf.d	
Additional .ini files parsed	/etc/php5/apache2/conf.d/05-opcache.ini, /etc/php5/apache2/conf.d/10-pdo.ini, /etc/php5/apache2/conf.d/20-gd.ini, /etc/php5/apache2/conf.d/20-json.ini, /etc/php5/apache2/conf.d/20-mcrypt.ini, /etc/php5/apache2/conf.d/20-mysql.ini, /etc/php5/apache2/conf.d/20-mysqli.ini, /etc/php5/apache2/conf.d/20-pdo_mysql.ini, /etc/php5/apache2/conf.d/20-readline.ini	
PHP API	20131106	
PHP Extension	20131226	
Zend Extension	220131226	
Zend Extension Build	API20131226,NTS	
PHP Extension Build	API20131226,NTS	
Debug Build	no	
Thread Safety	disabled	
Zend Signal Handling	disabled	
Zend Memory Manager	enabled	
Zend Multibyte Support	provided by mbstring	
IPv6 Support	enabled	
DTrace Support	enabled	
Registered PHP Streams	https, ftps, compress.zlib, compress.bzip2, php, file, glob, data, http, ftp, phar, zip	
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, sslv3, tls, tlsv1.0, tlsv1.1, tlsv1.2	
Registered Stream Filters	zlib.*, bzip2.*, convert.iconv.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk, mcrypt.*, mdecrypt.*	

Figura 52: funcionamiento de la página test.php en el servidor Apache



Figura 53: PHPMyAdmin

Una vez tenemos instalado de forma correcta MySQL, tendremos que cambiar el puerto por el que accede, ya que el puerto por defecto 3306 en la universidad está restringido para los usuarios ajenos a estas configuraciones del centro. Para ello, modificaremos en el archivo **my.cnf** el puerto cliente y el puerto de “**mysqld**”, así como la **bind-address** a **0.0.0.0**:

➤ **nano /etc/mysql/my.cnf**

9.2. ANEXO II: Instalación de MRBS

9.2.1. Introducción

Como ya se indicó en el apartado 5.4, **MRBS** es una sencilla aplicación web que consiste en un sistema de reserva de salas y gestión de horarios, idónea para gestionar lugares de reunión, asociaciones, o edificios con salas que necesiten de una reserva. Para su instalación, una vez tenemos configurado todos los requisitos necesarios (Apache, PHP y MySQL), debemos descargar los archivos de MRBS en el servidor web y crear la base de datos [83].

En primer lugar, tendremos que crear la base de datos y descargar los archivos de MRBS para desplegarlos en Apache. Los comandos a ejecutar en el terminal como superusuario son los siguientes:

```
➤ mysql -u root -p
➤ create database mrbs;
➤ exit;
➤ wget
https://sourceforge.net/projects/mrbs/files/latest/download?source=files
➤ tar -xvzf download?source=files
➤ cp -r mrbs-1.5.0/ /var/www/html
➤ cd /var/www/html
➤ mv mrbs-1.5.0 mrbs
➤ cd mrbs
➤ mysql -u root -p mrbs < tables.my.sql
➤ mysql -u root -p
➤ create user 'mrbs'@localhost' identified by 'mrbspi';
➤ grant all on mrbs.* to 'mrbs'@'localhost';
➤ Establecemos los parámetros de la base de datos de usuario (mrbs), contraseña (mrbspi), y zona horaria ($timezone="Europe/London").
nano /var/www/mrbs/web/config.inc.php
```

Para acceder a la aplicación web MRBS que acabamos de instalar, iremos a la siguiente dirección, poniendo la IP de nuestro servidor, por ejemplo:

<http://158.49.90.134/mrbs/web/index.php>

Ahora, para poder conectarnos a la base de datos de MRBS que se encuentra en la Raspberry, nos creamos un usuario con todos los privilegios para que la aplicación web que creemos pueda acceder a ella de manera remota:

```
➤ create user 'mrbs_remoto'@'%' identified by 'mrbspiremoto';
➤ grant all privileges on *.* to 'mrbs_remoto'@'%';
```

Figura 54: aplicación MRBS

9.2.2. Añadir gestor de tutorías

Con parte del Trabajo Fin de Grado de Jorge Alvarado tenemos la posibilidad también de gestionar las tutorías de los profesores en la misma aplicación de MRBS. Para incorporarla, solo hace falta pasar su modificación a nuestro MRBS:

➤ **`sudo cp -r Gestor\ Tutorias/* /var/www/html/mrbs/web/`**

Figura 55: aplicación MRBS con gestor de Tutorías

9.2.3. Establecer una dirección IP fija

Para terminar este anexo, explicaremos cómo podemos establecer a nuestro servidor, en nuestro caso la Raspberry Pi 2, una **dirección IP fija**. Con ello conseguimos que nuestra aplicación web externa al servidor pueda acceder a la base de datos MRBS de manera remota y sin necesidad de cambiar la dirección siempre que esta cambie [84].

Lo primero de todo, accedemos a nuestro router por **192.168.1.1** y contraseña admin – admin, o 1234 – 1234 (genéricas). Ahí vamos a la **configuración de NAT** y entramos el siguiente puerto de MySQL y dirección fija para la Raspberry Pi:

NAT - Edit SUA/NAT Server Set

	Start Port No.	End Port No.	IP Address
1	All ports	All ports	0.0.0.0
2	8081	8081	192.168.1.200

Figura 56: configuración NAT para dirección IP fija

Mediante el gestor de DNS dinámico **NO-IP** [85], creamos un dominio gratuito para la Raspberry, de tal forma que si ponemos esa ruta en el conector JDBC de MySQL de nuestra aplicación web, se podrá acceder remotamente sin variar el nombre de ruta. Usaremos el usuario creado **mrbs_remoto**, abierto a conexiones exteriores (%).

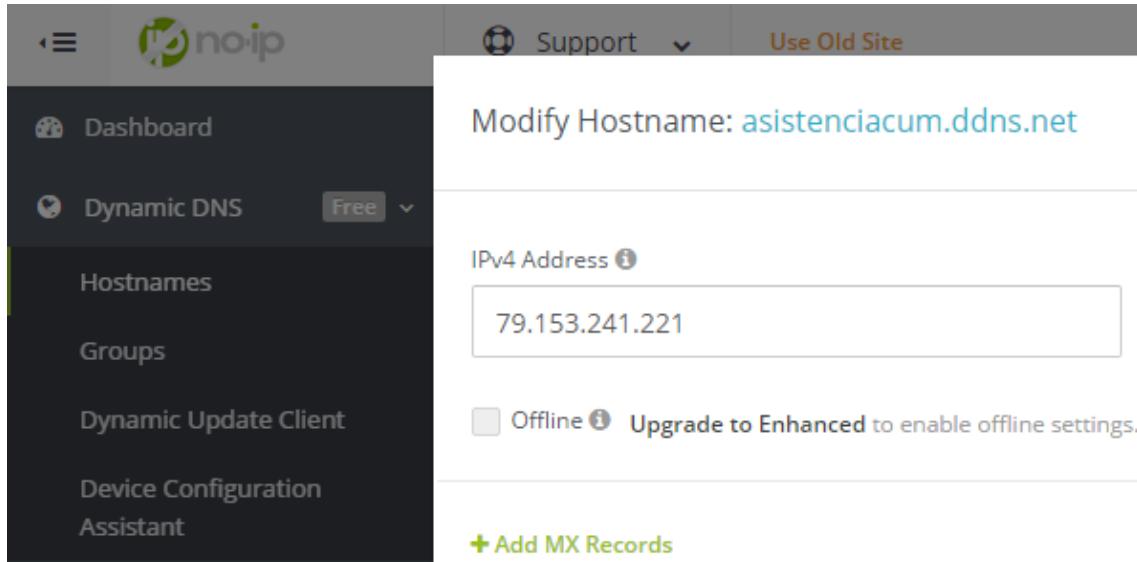


Figura 57: creación de dominio en NO-IP

```
***** 1. row *****
    Host: %
    User: mrbs_remoto
    Password: *AF69EA5C925C9B6FD158FE83A12749FB7CDF2166
    Select_priv: Y
    Insert_priv: Y
    Update_priv: Y
    Delete_priv: Y
    Create_priv: Y
    Drop_priv: Y
    Reload_priv: Y
    Shutdown_priv: Y
    Process_priv: Y
```

Figura 58: privilegios del usuario mrbs_remoto (permite conexiones exteriores)

En el **context.xml** de nuestro proyecto web se configura la conexión con la base de datos de MySQL:

```
<!-->
<Resource name="jdbc/testdb" auth="Container" type="javax.sql.DataSource"
  maxActive="10" maxIdle="5" maxWait="-1" username="mrbs_remoto" password="mrbspiremoto"
  driverClassName="com.mysql.jdbc.Driver" url="jdbc:mysql://asistenciacum.ddns.net:8081/mrbs" />
</Context>
```

Figura 59: context.xml que conecta con la base de datos MRBS de la Raspberry Pi 2

Este anexo lo cerramos facilitando los comandos para establecer una dirección IP estática y la instalación de NO-IP en la Raspberry:

- **sudo nano -w /etc/network/interfaces**
- Cambiamos la línea “**iface eth0 inet dhcp**” o “**iface eth0 inet manual**” (dependiendo de la versión del sistema) y establecemos la IP que queramos:
 - **auto eth0**
 - **iface lo inet loopback**
 - **iface eth0 inet static**
 - **address 192.168.1.75**
 - **netmask 255.255.255.0**
 - **gateway 192.168.1.1**
- NO-IP se instala con lo siguiente:
 - **mkdir noip**
 - **cd noip**
 - **wget <http://www.no-ip.com/client/linux/noip-duc-linux.tar.gz>**
 - **tar zxvf noip-duc-linux.tar.gz**
 - **cd noip-2.1.9-1**
 - **sudo apt-get install build-essential**
 - **make**
 - **sudo make install**

9.3. ANEXO III: Arduino y lector de huellas Adafruit

9.3.1. Introducción

En la sección 5.2 indicamos el uso de la placa microcontroladora de **Arduino UNO** y que se combinaría con el **lector de huellas de Adafruit**. En este anexo vamos a dar los pasos para conectar el Arduino y el sensor, además de la **pantalla LCD**.

9.3.2. Instalación de Arduino IDE

Arduino tiene su propio entorno de desarrollo de código abierto, **IDE Arduino**, basado en el entorno de *Processing* y lenguaje de programación basado en *Wiring*. Su instalación es muy simple y se puede obtener a través del siguiente enlace: [86].

Este entorno de Arduino incorpora ya ejemplos desarrollados para ponerlos a prueba con las diferentes extensiones que se pueden conseguir de esta placa a través de la conexión de distintos dispositivos.

9.3.3. Instalación de la librería Adafruit

En cuanto al lector de huellas, Adafruit nos ofrece una librería que podemos descargar aquí [87], y que debemos de agregar a la carpeta de librerías que contiene todos los ejemplos el IDE Arduino (en la carpeta donde se haya instalado el IDE, accedemos a la carpeta “*libraries*”).

Una vez insertada la librería, esta ya nos debe aparecer en el entorno de desarrollo, con los ejemplos de registrar huella, eliminar huellas, mostrar *templates*, etc.:

- **Enroll.** En este ejemplo se desarrolla el programa con el cual podremos almacenar nuevas huellas en el lector. Requiere dos imágenes de la huella a registrar.
- **Fingerprint.** Comprueba que una huella esté almacenada. Es el programa con el que se procede a la identificación de una persona, devolviendo el identificador de la huella encontrada.
- **Show_fingerprint_templates.** Recupera e imprime los datos de las plantillas: la memoria está organizada en “**páginas**”. Cada una contiene los datos de una única huella, llamado *template* o plantilla. Para leer una plantilla:
 1. Se transfiere la plantilla al búfer de caracteres del lector de huellas.
 2. Se transfiere el búfer de caracteres al ordenador y se muestran los datos.
- **Blank.** Programa vacío con el que podemos verificar la conexión con el Arduino.
- **Delete.** Para borrar alguna huella almacenada haremos uso de este ejemplo. Para ello se ha de introducir el identificador de la huella que queremos eliminar.
- **Leo_passthru.** Permite a un **Arduino Leonardo** pasar datos en serie entre el lector de huellas y Windows.

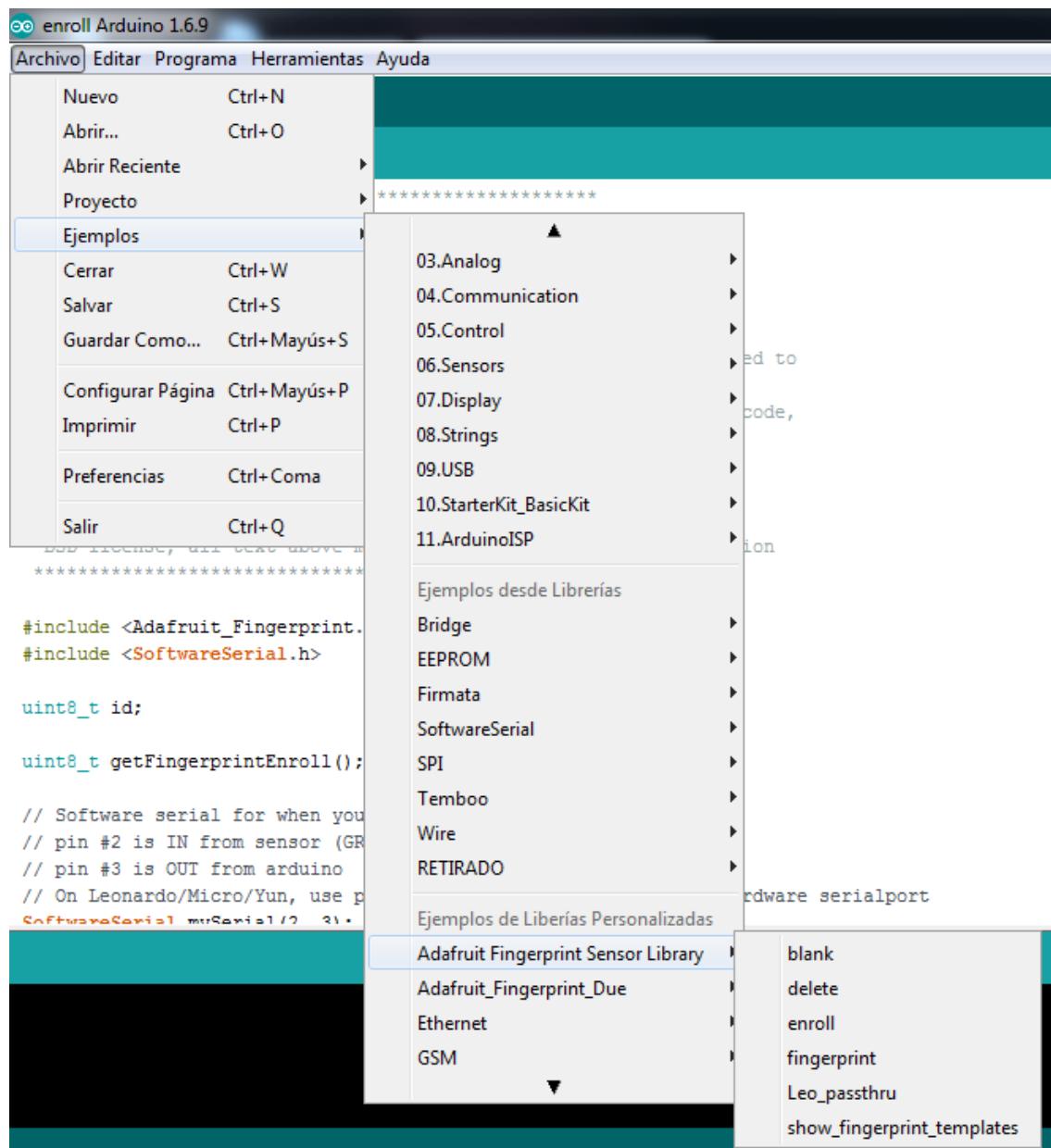


Figura 60: ejemplos de la librería del sensor de huellas Adafruit

También Adafruit nos ofrece una interfaz de usuario a través del programa **SFG Demo**, por el que podemos ejecutar todos los ejemplos anteriores de manera interactiva y más visual, pero su funcionamiento es muy irregular [88].

9.3.4. Conexiones entre el Arduino, el lector de huellas y la pantalla LCD

Para poder ejecutar los ejemplos de la librería, es necesario tener conectado el sensor de huellas al Arduino. Al ser de **Grove**, la conexión no tiene complicaciones, y solo tendremos que conectar el cable del Arduino al lector.

Este cable está compuesto de cuatro pines de colores diferentes:

- **Negro.** Conexión a tierra (GND).
- **Rojo.** Alimentación de 5 voltios.
- **Amarillo.** Salida de datos.
- **Blanco.** Entrada de datos.

Una vez conectado el lector al microcontrolador por medio del puerto **digital 2**, debemos de dar alimentación al Arduino con un cable USB [89].



Figura 61: sensor de huellas conectado a una placa Grove, que a su vez se integra en un Arduino UNO

Las **pantallas LCD** [90], que suelen venir incorporadas en los kits de iniciación con el Arduino Grove, la única objeción que tienen es que deben ser conectadas a algún puerto de **comunicación I2C** de la placa Grove.

Las LCD también tienen su propia librería de ejemplos: *LiquidCrystal* o la propia de Grove, **Grove_LCD_RGB**.



Figura 62: pantalla Grove-LCD RGB Backlight

9.3.5. Conexión Java - Arduino

Como se indicó en el apartado 6.1.3, Arduino se conecta con nuestra aplicación Java a través del puerto serie. Esto se posibilita gracias al uso de una librería que permite la conexión entre Java y Arduino, para así tener la funcionalidad de pasar datos e información de un lado a otro, básicamente para poder indicar al lector de huellas qué acción realizar cuando el profesor lo indique a través de la web.

Esta librería está desarrollada por el grupo del sitio web llamado “**Panamá Hitek**”, y que viene perfectamente explicada y descrita aquí [91].

En nuestro proyecto, ya dijimos en el apartado anteriormente mencionado que esta librería la desplegamos en una clase llamada **ConexionArduino.java**, dentro del paquete **Util**. Para ello, debemos incorporar el archivo de extensión JAR con nombre **RXTXcomm.jar** en la carpeta de Java **/jre/lib/ext**, mientras que la biblioteca de enlace dinámico o DLL llamado **rxtxSerial.dll** se debe añadir a la carpeta **/jre/bin**. De esta forma podemos ya importar la librería a nuestro proyecto [92].

9.3.6. Puerto de conexión Arduino

En cuanto al puerto de conexión de Arduino, este varía de un ordenador a otro, y en el caso de que el Arduino cambie de situación, se debe modificar el nombre del puerto.

Esta función la pueden realizar los administradores desde la web en la opción “**Arduino**”, y como bien viene explicado en esa página, el nombre del puerto se puede ver en el **Administrador de dispositivos de Windows**, mientras que en una plataforma Linux (Debian, Ubuntu, Mac OS X...), es necesario ejecutar el comando: **dmesg | grep ttyACM**.

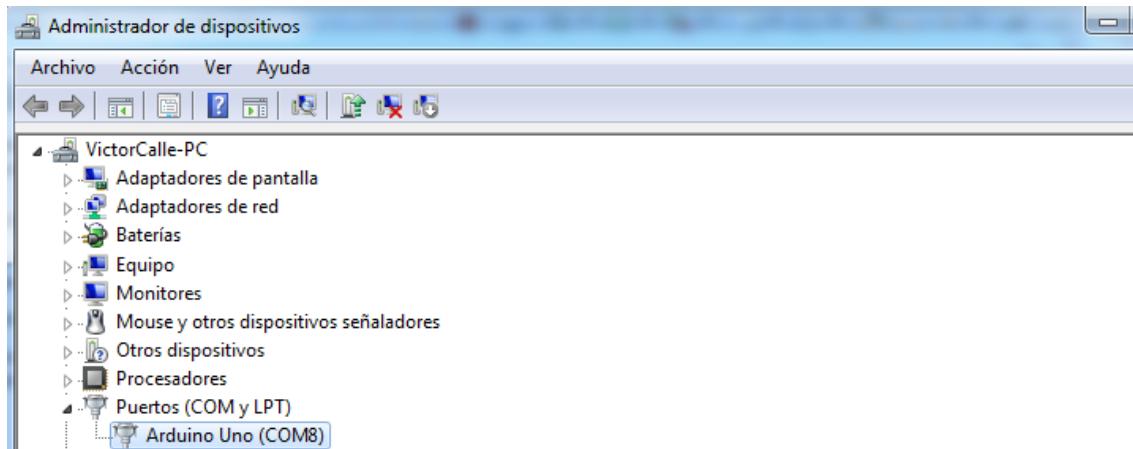


Figura 63: puerto Arduino en Windows

```
victorcalle@victorcalle-Lenovo-B560:~$ lsusb
Bus 002 Device 004: ID 0bda:0159 Realtek Semiconductor Corp. RTS5159 Card Reader Controller
Bus 002 Device 003: ID 0781:5583 SanDisk Corp.
Bus 002 Device 005: ID 2341:0043 Arduino SA Uno R3 (CDC ACM)
Bus 002 Device 002: ID 8087:0020 Intel Corp. Integrated Rate Matching Hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 006: ID 04f2:b273 Chicony Electronics Co., Ltd
Bus 001 Device 005: ID 1c7a:0801 LighTuning Technology Inc. Fingerprint Reader
Bus 001 Device 004: ID 0489:e00d Foxconn / Hon Hai Broadcom Bluetooth 2.1 Device
Bus 001 Device 003: ID 062a:3281 Creative Labs
Bus 001 Device 002: ID 8087:0020 Intel Corp. Integrated Rate Matching Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
victorcalle@victorcalle-Lenovo-B560:~$ dmesg | grep ttyACM
[    67.332488] cdc_acm 2-1.2:1.0: ttyACM0: USB ACM device
victorcalle@victorcalle-Lenovo-B560:~$
```

Figura 64: puerto Arduino en plataforma Linux

9.4. ANEXO IV: Jasper Reports

9.4.1. Introducción

La sección de implementación de la capa de presentación (6.1.1) ya explicó cómo se iban a realizar los informes: por medio de la biblioteca de creación de reportes **JasperReports Server**, la cual puede ser incrustada en cualquier aplicación Java. Procedemos aquí a mostrar los pasos a seguir para instalar esta biblioteca en **Eclipse**, la plataforma de software donde hemos desarrollado nuestro proyecto:

9.4.2. Instalación de Jasper Reports en Eclipse

1. Instalamos **Jaspersoft** en **Eclipse** arrastrando (*drag and drop*) el enlace de instalación que proporcionan en su web a nuestro entorno de desarrollo [93].

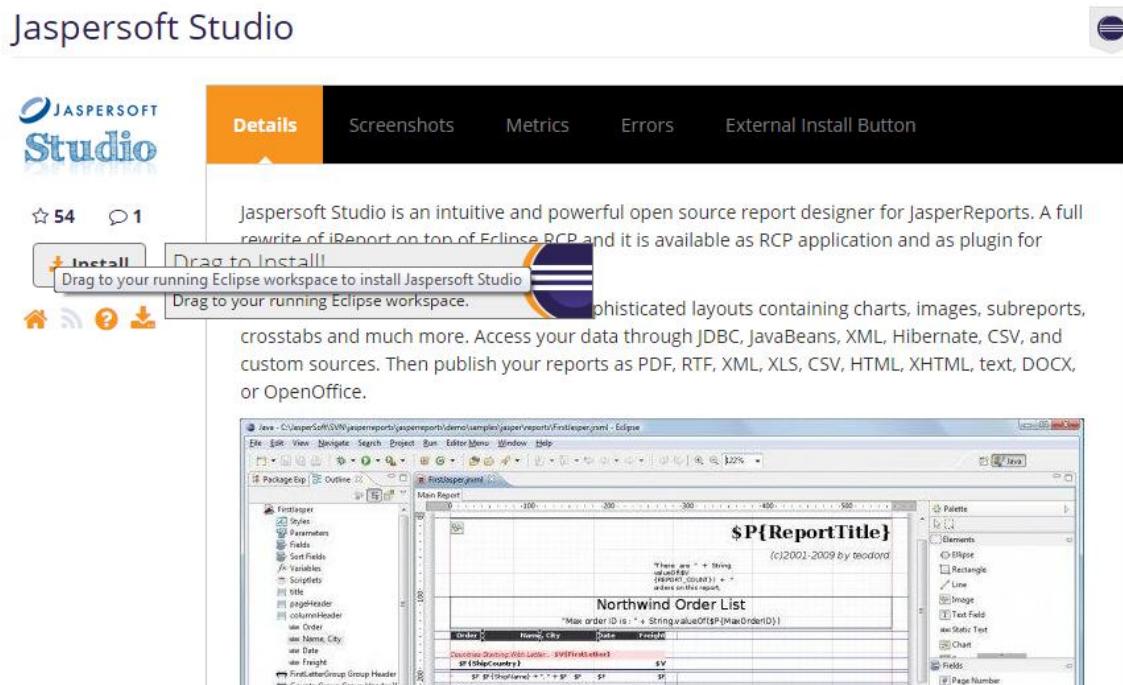


Figura 65: JasperSoft Studio

2. Una vez haya terminado de instalarse, abrimos el **Repository Explorer** y creamos un nuevo adaptador, de tipo “*Database JDBC Connection*”.

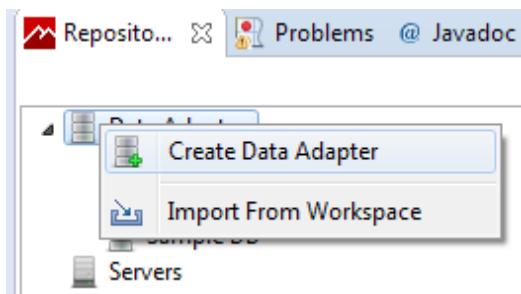


Figura 66: creación del adaptador

3. Damos un nombre al adaptador y elegimos el driver JDBC “**com.mysql.jdbc.driver**”, poniendo la dirección de la base de datos y el usuario y contraseña con el que accedemos a ella.

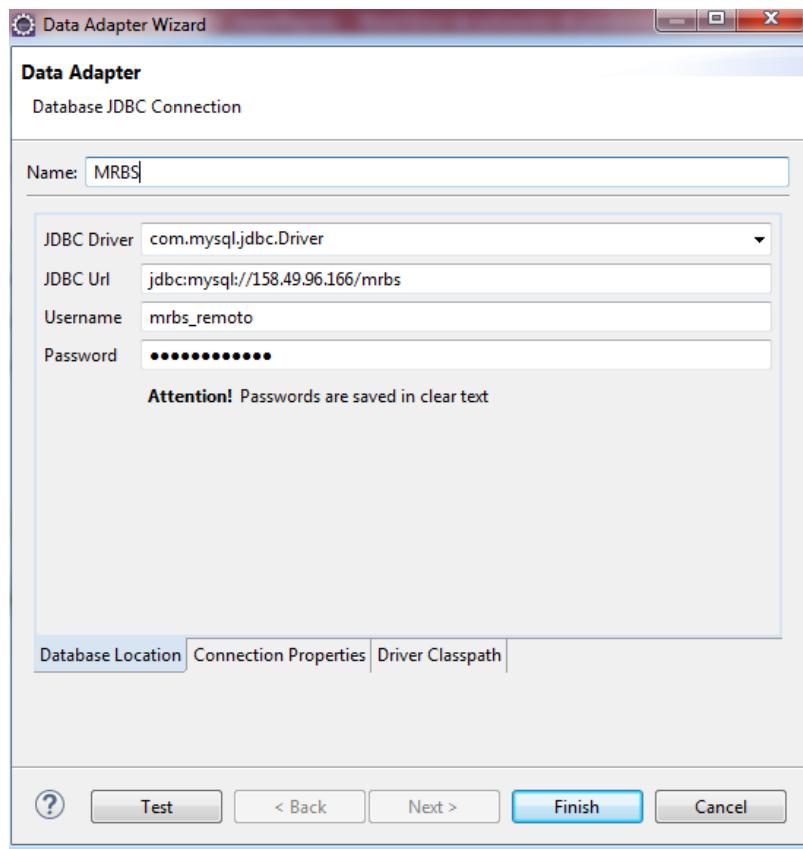


Figura 67: configuración de la conexión JDBC en el adaptador

9.4.3. Creación de informes con Jasper Reports

Para cada informe que deseemos generar, tenemos que crear un archivo con extensión **jxml**, ficheros con los que se llevan a cabo los reportes en Jasper Reports. La creación de un fichero de este tipo se realiza de la siguiente forma en Eclipse:

1. Creamos un nuevo fichero Jasper Report y elegimos la plantilla que queremos que nuestro informe tenga de diseño, además de darle un nombre al archivo jxml tras pulsar en “Siguiente”.

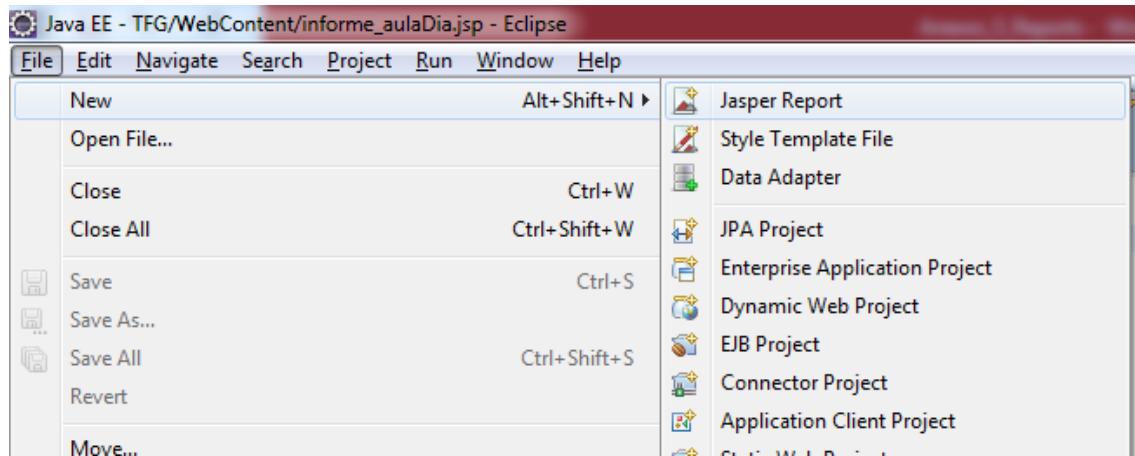


Figura 68: creación de fichero Jasper Report en Eclipse

2. Seleccionamos la base de datos de la que el reporte se va alimentar de datos, y escribimos la consulta con la que se muestra la información requerida.

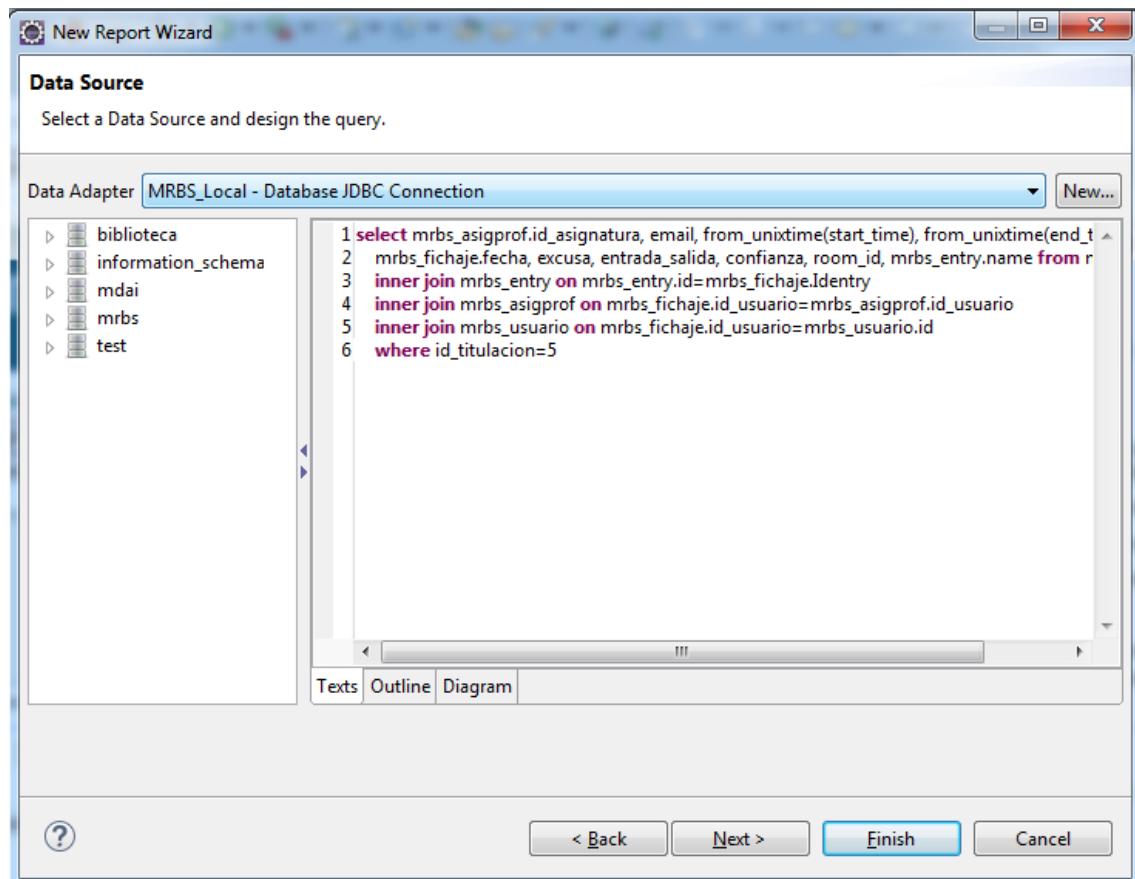


Figura 69: selección de la base de datos e introducción de la consulta que realizará el informe

3. Tras pulsar “Siguiente”, terminamos el reporte seleccionando los campos que queremos que se muestren en el informe PDF.

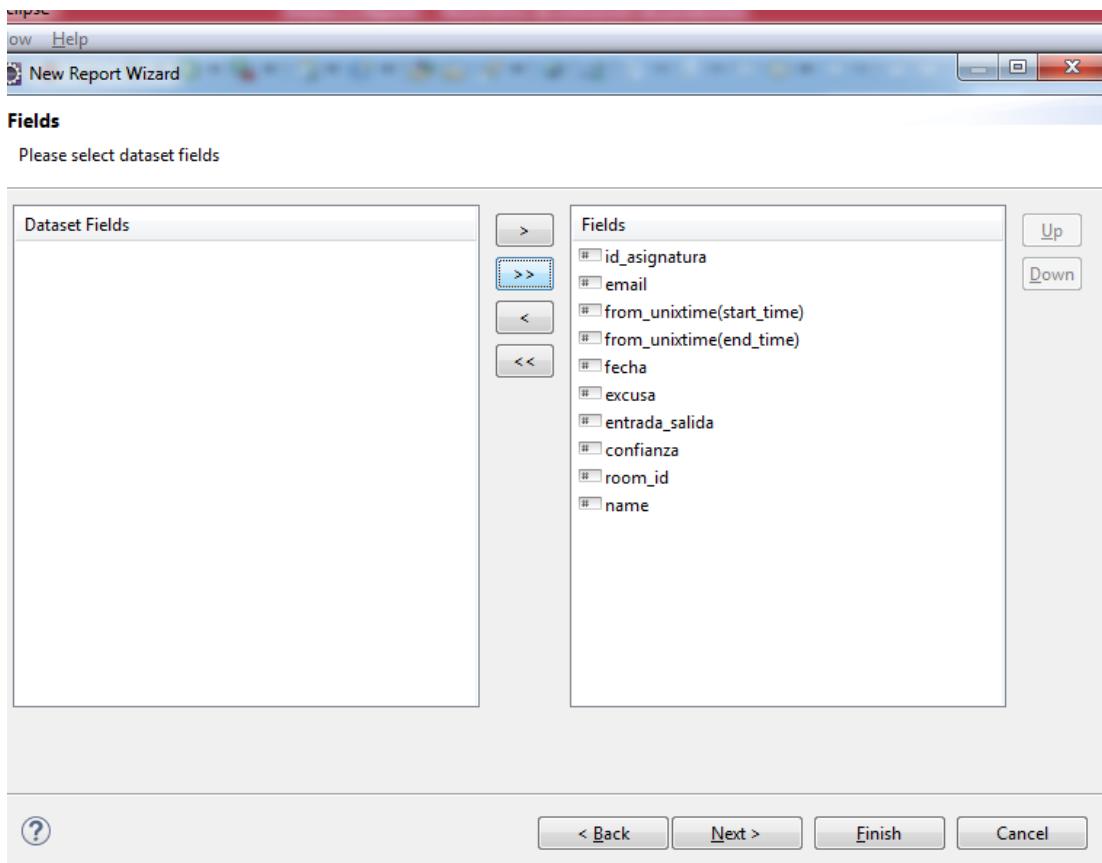


Figura 70: selección de los campos recuperados por la consulta

4. Ya tenemos creado el informe con los datos consultados, y se abrirá el editor de reportes, donde se pueden realizar múltiples acciones: editar la posición de la cabecera, añadir tablas, cambiar el formato de los textos, incorporar imágenes, modificar los estilos, poner un calendario o la fecha en la que se genera el PDF, etc.
5. Sólo vamos a explicar una de las opciones más importantes de los reportes: **añadir parámetros**. Estos parámetros servirán para poner condiciones en nuestras consultas, de tal forma que el parámetro será una variable que recibirá el dato cuando se llame al reporte. Sobre el menú “Parameters” pulsamos con el botón derecho y damos a crear un parámetro. Éste se añadirá a la lista y lo podremos editar nuevamente con el click derecho y eligiendo la opción “Show Properties” (ofrece cambiar el nombre, añadir una descripción, cambiar el tipo de dato...).

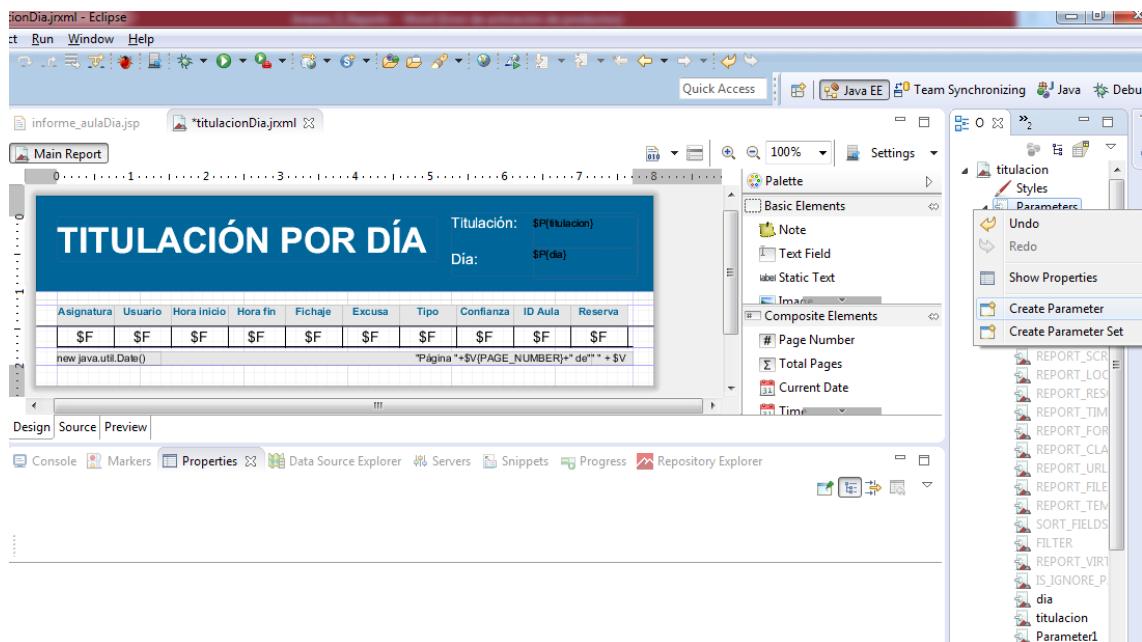


Figura 71: editor del informe Jasper Report

6. Por último, si queremos añadir este parámetro en nuestra consulta, tendremos que entrar en la opción “**DataSet and Query editor Dialog**” situado en la parte posterior del reporte cerca del zoom. Editamos la consulta y modificamos las condiciones para poner en su lugar el nombre del parámetro con el formato: **\$P{nombreParametro}**. Los parámetros también pueden incorporarse a nuestro reporte para mostrar su valor, simplemente pinchando sobre el parámetro y arrastrándolo a la posición en la que deseamos que aparezca en el informe.

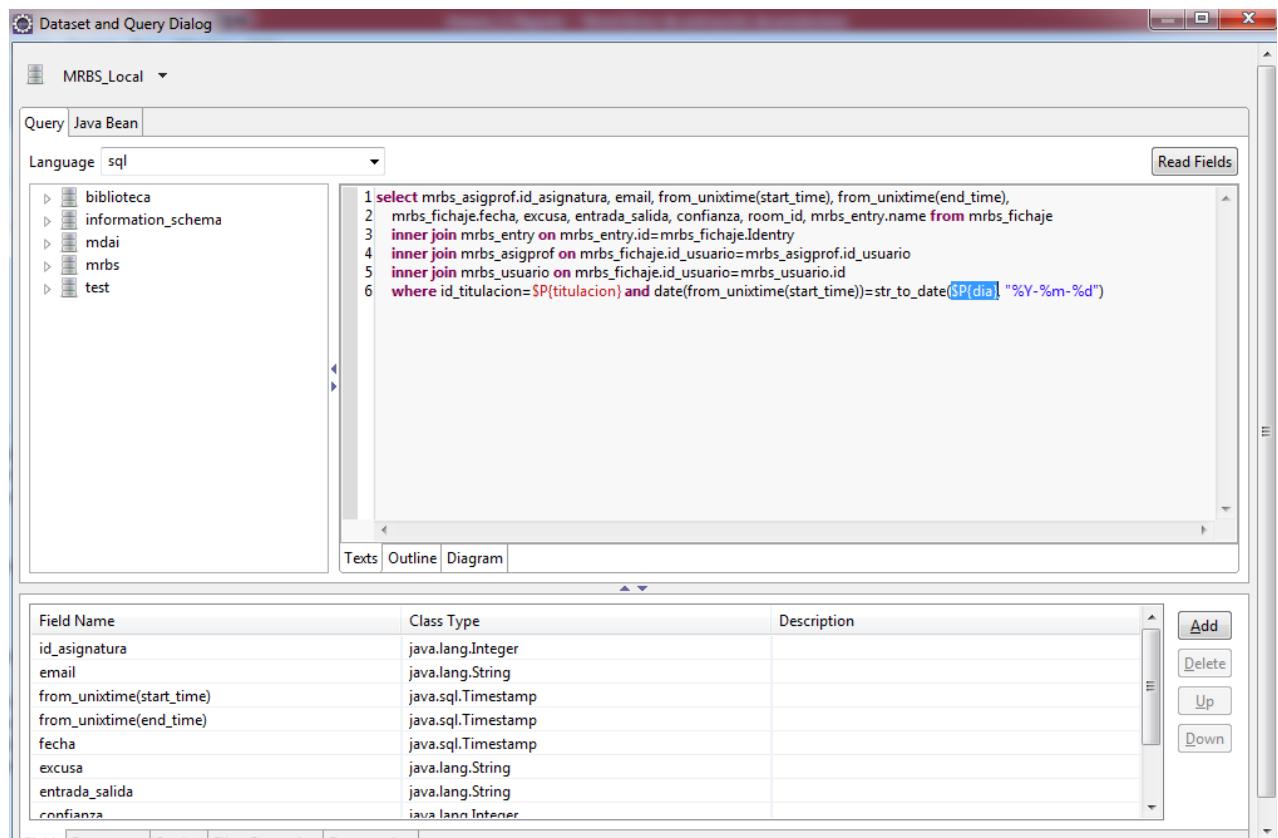


Figura 72: incorporación de parámetros en una consulta

Estos archivos de extensión `.jrxml` se encuentran en una carpeta del “*WebContent*” bajo el nombre “**Reportes**”. Ahora, para que estos reportes se puedan utilizar desde la página web, debemos crear para cada uno de ellos sus correspondientes JSP:

1. En cada JSP es importante indicar en la cabecera que el contenido será de tipo PDF:


```
<%@ page contentType="application/pdf"%>
```
2. Se abre una conexión con la base de datos a través de la clase `ConexionUtil.java` por medio del conector JDBC.
3. Se obtiene la lista de parámetros que se han indicado en el previo formulario, y estos se hacen corresponder con los parámetros del informe.
4. En estos archivos JSP solo varía de uno a otro los nombres de los parámetros obtenidos del formulario correspondiente, y el nombre del reporte `.jrxml` al que se atribuye.

```
<%@ page contentType="application/pdf"%>
<%@ page import="net.sf.jasperreports.engine.*"%>
<%@ page import="java.util.*"%>
<%@ page import="java.io.*"%>
<%@ page import="es.unex.cum.tfg.model.*"%>
<%@ page import="util.*"%>
<%@ page import="java.sql.Connection"%>
<title>Aula por día</title>
<%
try {
    ConexionUtil conexion = new ConexionUtil();

    Map<String, Object> parameters = new HashMap<String, Object>();
    ArrayList<String> parametros = (ArrayList<String>) request
        .getAttribute("parametros");
    parameters.put("aula", parametros.get(1));
    parameters.put("dia", parametros.get(0));
    String jrxmFile = session.getServletContext().getRealPath(
        "Reportes/aulaDia.jrxml");
    InputStream input = new FileInputStream(new File(jrxmFile));
    JasperReport reporte = JasperCompileManager
        .compileReport(input);
    JasperPrint jasperprint = JasperFillManager.fillReport(reporte,
        parameters, conexion.getConexion());
    JasperExportManager.exportReportToPdfStream(jasperprint,
        response.getOutputStream());
    response.getOutputStream().flush();
    response.getOutputStream().close();
} catch (Exception e) {
    System.out.println(e.getMessage());
}
%>
```

Algoritmo 19: ejemplo de código en un JSP que embebe a un informe de JasperReport

9.5. ANEXO V: Despliegue de la aplicación web

9.5.1. Introducción

Para facilitar la ejecución de la aplicación por parte del usuario, haremos un explicativo para aprender a instalar y desplegar el proyecto a través de **XAMPP**, un sencillo entorno gráfico distribuido por Apache y que contiene **Apache Tomcat**, el servidor de aplicaciones que ya hemos dicho que usaremos para este proyecto [94].

9.5.2. Instalación de XAMPP

La instalación de XAMPP es muy fácil y sencilla. Tan solo hay que descargar el instalador para nuestro sistema operativo [95] (muy importante que sea la última versión del programa), y procedemos a ejecutar el archivo descargado.

El único aspecto a destacar en cuanto al desarrollo de la instalación es que debemos de permanecer seleccionado los componentes de servidor **Apache** y **Tomcat**, es decir, en el caso de Windows no deberíamos deseleccionar dichos componentes del apartado “*Server*”:

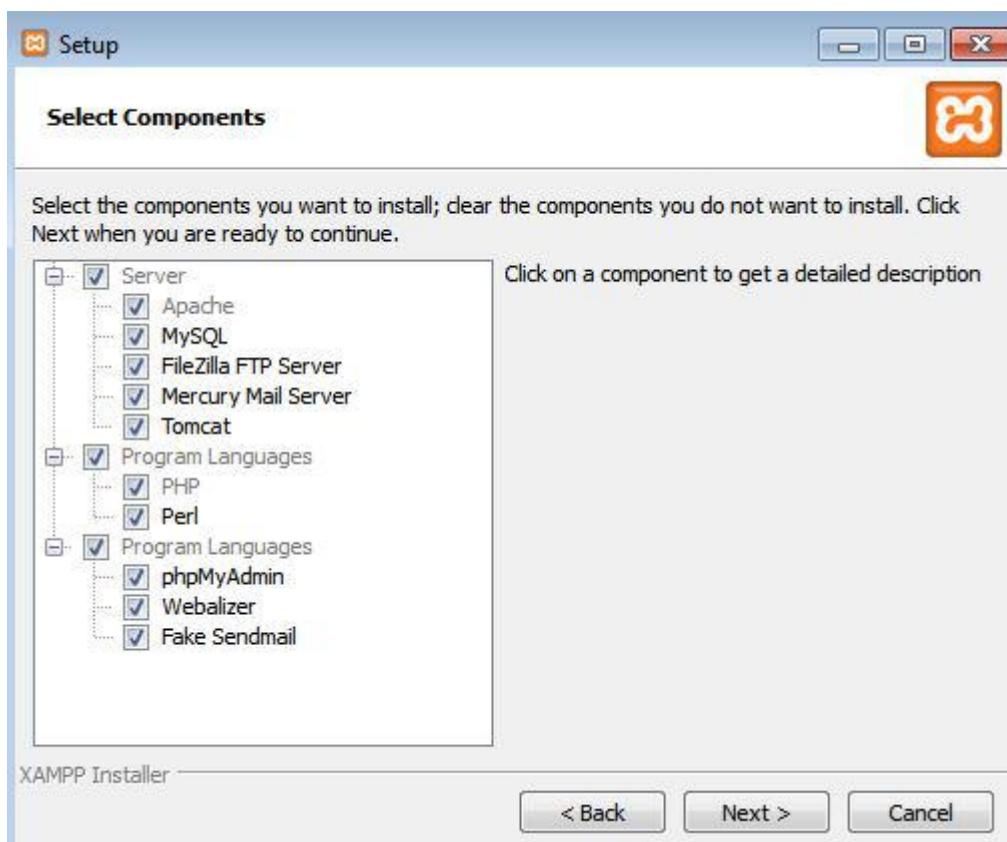


Figura 73: instalador de XAMPP

Una vez finalizada la instalación, tendremos que añadir un usuario “*manager*” a Tomcat. Pulsamos sobre el botón “*config*” de Tomcat y abrimos el archivo “**tomcat-users.xml**”, para incluir las siguientes líneas:

```
<role rolename="manager-gui"/>
<user username="tomcat" password="controlasistenciacum"
roles="manager-gui"/>
```

```

tomcat-users: Bloc de notas
Archivo Edición Formato Ver Ayuda
<?xml version='1.0' encoding='utf-8'?>
<!--
Licensed to the Apache Software Foundation (ASF) under one or more
contributor license agreements. See the NOTICE file distributed with
this work for additional information regarding copyright ownership.
The ASF licenses this file to You under the Apache License, Version 2.0
(the "License"); you may not use this file except in compliance with
the License. You may obtain a copy of the License at
    http://www.apache.org/licenses/LICENSE-2.0
Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
-->
<tomcat-users>
<!--
NOTE: By default, no user is included in the "manager-gui" role required
to operate the "/manager/html" web application. If you wish to use this app,
you must define such a user - the username and password are arbitrary.
-->
<!--
NOTE: The sample user and role entries below are wrapped in a comment
and thus are ignored when reading this file. Do not forget to remove
<!... ...> that surrounds them.
-->
<role rolename="manager-gui"/>
<user username="tomcat" password="controlasistenciacum" roles="manager-gui"/>
</tomcat-users>

```

Figura 74: fichero de configuración tomcat-users.xml

Ya podemos ir arrancando Apache y Tomcat. El único requisito previo para que estos componentes puedan ejecutarse es tener previamente instalado **Java JDK o JRE**, algo que ya es imprescindible para nuestros ordenadores en estos tiempos y que sin duda ya tendremos instalado. En caso contrario, accedé al siguiente enlace y descargad la última versión del kit de desarrollo Java para vuestro sistema operativo [96].

En las carpetas generadas al instalar Java, debemos de introducir unos *drivers* en el JRE que harán posible el uso de la librería **RXTX** (`C:\Program Files\Java\jre1.8.0_91`). Es decir, el archivo **rxtxSerial.dll** tendremos que copiarlo a la carpeta bin de dicha ruta, mientras que en `\lib\ext\` incorporamos **RXTXcomm.jar**. Estos ficheros los podéis obtener en este enlace [97].

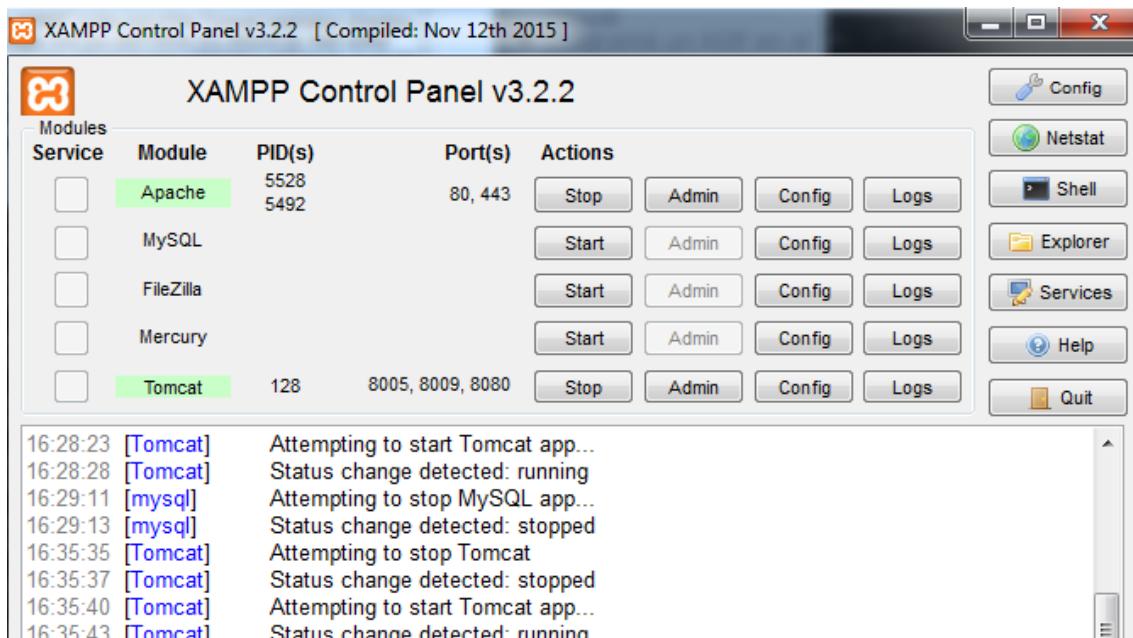


Figura 75: interfaz gráfica de XAMPP

Entramos en <http://localhost:8080/> y accedemos a la opción “Manager App”. Nos pedirá credenciales, que serán las que hemos puesto en el fichero para el usuario creado anteriormente.

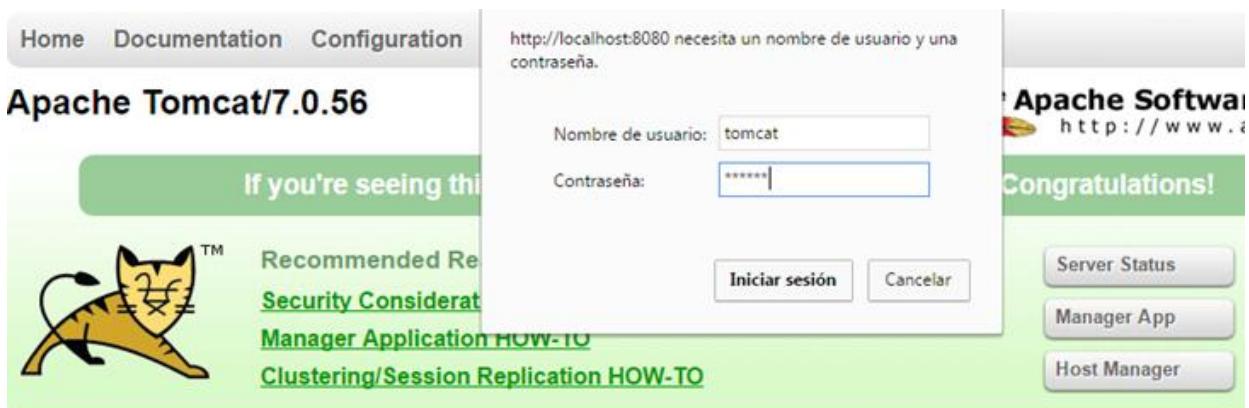


Figura 76: Apache Tomcat

Ahora desplegamos el archivo .WAR que contiene el proyecto “ControlAsistenciaCUM”.

Desplegar
Desplegar directorio o archivo WAR localizado en servidor
<input type="text" value="Trayectoria de Contexto (opcional):"/> <input type="text" value="URL de archivo de Configuración XML:"/> <input type="text" value="URL de WAR o Directorio:"/> <input type="button" value="Desplegar"/>
Archivo WAR a desplegar
Seleccione archivo WAR a cargar <input type="button" value="Seleccionar archivo"/> ControlAsistenciaCUM.war <input type="button" value="Desplegar"/>

Figura 77: realización de despliegues en Apache Tomcat

Y si todo ha funcionado correctamente, aparecerá un mensaje de “OK” en la parte superior de la página. Ya la tenemos desplegada en la lista de aplicaciones de Apache Tomcat.



The Apache Software Foundation
http://www.apache.org/

Gestor de Aplicaciones Web

Mensaje:	OK		
Gestor			
Listar Aplicaciones	Ayuda HTML de Gestor		
Aplicaciones			
Trayectoria	Versión	Nombre a Mostrar	Ejecutándose
/	Ninguno especificado	Welcome to Tomcat	true
/ControlAsistenciaCUM	Ninguno especificado	TFG	true
/docs	Ninguno especificado	Tomcat Documentation	true

Figura 78: lista de aplicaciones desplegadas en Apache

La URL para acceder a la aplicación web será:

<http://localhost:8080/ControlAsistenciaCUM/>