



Universidad de Valladolid

E.T.S Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática de Sistemas

**Actualización de una aplicación web y móvil para
el reconocimiento de usuario por voz**

Autor:

D. Héctor Pablos López

Tutor:

D. Carlos Enrique Vivaracho Pascual

Resumen

Los sistemas de reconocimiento biométrico tienen una serie de aplicaciones prácticas, en todos los aspectos de la sociedad, que pueden llegar a formar parte de nuestra vida diaria, haciendo que tareas rutinarias, como la autenticación o la identificación, se puedan realizar de forma más rápida, segura y cómoda.

Este trabajo se centra en llevar el reconocimiento biométrico por voz a los usuarios, en formas que sean fáciles de utilizar y ampliamente accesibles, como las páginas web y las aplicaciones móviles. Para ello, se ha desarrollado un sistema de verificación de locutor, partiendo de los requisitos de usuario de un sistema anterior, que pretende seguir las buenas prácticas de programación conocidas a día de hoy: Documentación exhaustiva, código limpio y bien estructurado, utilización del paradigma de orientación a objetos, patrón modelo vista controlador, gestión de errores mediante excepciones y tests unitarios. Así como un diseño que siga unos criterios de calidad establecidos, como reutilización, portabilidad, configurabilidad y extensibilidad, alcanzados mediante el uso de patrones arquitectónicos y de diseño.

Abstract

Biometric authentication systems have a series of practical applications, in every aspect of our society, that may make them part of our daily lives, making routinary tasks, as authentication and identification, faster, safer and easier to perform.

This work focuses on taking biometric authentication to users, in easy to use and widely accessible ways, such as web pages and mobile applications. In order to achive this, a speaker verification system has been developed, from the user requisites of an earlier project, by following up to date known best programming practices such as exhaustive documentation, clean and well structured code, object oriented paradigm usage, model view controller pattern, error handling with exceptions and unit tests. As well as a design that complies with the established quality standars, suh as reusability, portability, configurability and extensibility, achieved by using architectural and design patterns.

Agradecimientos

A Carlos Enrique Vivaracho, por su inestimable colaboración en la realización del proyecto. No por típica es menos valiosa la aparición del tutor en este apartado.

Por extensión, a todos los profesionales dedicados a la enseñanza pública.

A Cifu, Natalia y Pato, por hacer que el camino haya sido mucho más fácil y divertido.

A mi familia y amigos.

Índice general

1. Introducción	1
1.1. Descripción del proyecto	1
1.1.1. Metodología utilizada	1
1.2. Objetivos del proyecto	2
1.2.1. Objetivo general	2
1.2.2. Objetivos específicos	2
1.3. Estructura del documento	3
2. Estado del arte	5
2.1. Introducción	5
2.2. Reconocimiento de locutor	6
2.3. Verificación de locutor independiente de texto	7
2.3.1. Parametrización de la muestra de audio	7
2.3.2. Modelado estadístico	10
2.3.3. Normalización	11
2.3.4. Evaluación	12
2.4. Software para el reconocimiento de locutor	13
2.4.1. Tratamiento de audio	13
2.4.2. Extracción de características	13
2.4.3. Verificación de locutor	13
2.5. Pruebas con Alize	15
2.5.1. Datos del corpus	15
2.5.2. Procedimientos involucrados en las pruebas	15
2.5.3. Parámetros de las pruebas	16
2.5.4. Resultados de las pruebas	17
2.5.5. Conclusiones	20
3. Plan de desarrollo	21
3.1. Introducción	21
3.2. Vista general del proyecto	22
3.2.1. Propósitos, alcance y objetivos	22

3.2.2.	Supuestos y restricciones	22
3.2.3.	Entregables del proyecto	22
3.2.4.	Resumen de la programación y el presupuesto	23
3.2.5.	Evolución del plan	23
3.3.	Organización del Proyecto	25
3.3.1.	Interfaces externas	25
3.3.2.	Estructura interna	25
3.3.3.	Roles y responsabilidades	25
3.3.4.	Métodos de trabajo	27
3.4.	Proceso de gestión	28
3.4.1.	Puesta en marcha del plan	28
3.4.2.	Plan de trabajo	28
3.4.3.	Plan de control del proyecto	37
3.4.4.	Plan de riesgos	38
3.4.5.	Plan de cierre	45
3.5.	Proceso técnico	46
3.5.1.	Métodos, herramientas y técnicas	46
4.	Análisis	49
4.1.	Introducción	49
4.1.1.	Propósito del sistema	49
4.1.2.	Planteamiento del problema y objetivos	49
4.2.	Especificación de requisitos	53
4.2.1.	Requisitos de usuario	53
4.2.2.	Requisitos funcionales	53
4.2.3.	Requisitos no funcionales	53
4.2.4.	Requisitos de información	55
4.3.	Casos de uso	56
4.3.1.	Diagrama de casos de uso	56
4.3.2.	CU01 - Registro en el sistema	56
4.3.3.	CU02 - Inscripción en el sistema	58
4.3.4.	CU03 - Verificación de locutor	60
4.4.	Matriz de trazabilidad	62
4.5.	Modelo de dominio	63
4.6.	Diagramas de secuencia	64
4.6.1.	CU01 - Registro de usuario	64
4.6.2.	CU02 - Inscripción en el sistema	65
4.6.3.	CU03 - Verificación de locutor	66

5. Diseño	67
5.1. Introducción	67
5.2. Objetivos arquitectónicos y filosofía	68
5.2.1. Reutilización	68
5.2.2. Extensibilidad	68
5.2.3. Configurabilidad	68
5.2.4. Portabilidad	69
5.3. Arquitectura del sistema	70
5.3.1. Visión global	70
5.3.2. Diseño de la arquitectura	75
5.4. Patrones	79
5.4.1. Adaptador (Wrapper)	79
5.4.2. Fachada	80
5.4.3. Método factoría	80
5.5. Casos de uso de diseño	83
5.5.1. CU01 - Registro en el sistema	83
5.5.2. CU02 - Inscripción en el sistema	84
5.5.3. CU03 - Verificación de locutor	86
5.6. Diagramas de clases y de secuencia	89
6. Implementación	95
6.1. Introducción	95
6.2. Elementos relevantes en el desarrollo	96
6.2.1. Servidor	96
6.2.2. IDEs y software de pruebas	96
6.2.3. Framework PHP	96
6.2.4. Librerías Android	97
6.2.5. Documentación interna	97
6.3. Control de versiones y gestión de dependencias	99
6.4. Formación	100
7. Pruebas	101
7.1. Introducción	101
7.2. Plan de pruebas	102
7.3. Pruebas unitarias: alizephp	103
7.3.1. Casos de prueba	103
7.4. Pruebas unitarias y de integración: BioVoiceAPI	110
7.4.1. Registro de usuario	110
7.4.2. Inscripción de locutor	117

7.4.3. Verificación de locutor	118
7.5. Pruebas de integración: BioVoiceWeb y BioVoiceAPP	121
8. Control, seguimiento y plan de cierre	123
8.1. Introducción	123
8.2. Objetivos conseguidos	124
8.3. Control y seguimiento del proyecto	125
8.3.1. Control de la programación temporal del proyecto	125
8.3.2. Artefactos obtenidos	126
8.3.3. Reuniones	126
8.3.4. Problemas encontrados	127
9. Conclusiones y trabajo futuro	129
9.1. Introducción	129
9.2. Conclusiones	130
9.3. Trabajo futuro	132
9.3.1. Identificación de usuarios	132
9.3.2. Mejora de resultados de verificación de locutor	132
Bibliografía	133
A. Contenido del CD	135
B. Versiones de este documento	137

Índice de figuras

2.1. Fase de entrenamiento en verificación de locutor	7
2.2. Fase de test en verificación de locutor	7
2.3. Parametrización basada en filterbank	8
2.4. Transformaciones de ventana: Hamming, Hanning y rectangular	9
2.5. Parametrización basada en LPC	10
3.1. ISO/IEC 27005:2008: Estándar utilizado para identificar la exposición al riesgo . .	38
4.1. Diagrama de casos de uso	56
4.2. Modelo de dominio	63
4.3. Diagrama de secuencia CU01 - Registro de usuario	64
4.4. Diagrama de secuencia CU02 - Inscripción en el sistema	65
4.5. Diagrama de secuencia CU03 - Verificación de locutor	66
5.1. Diagrama de despliegue	72
5.2. Diagrama de componentes	73
5.3. Diagrama de capas	74
5.4. Diagrama de paquetes	76
5.5. Vista lógica del sistema	77
5.6. Patrón adaptador	79
5.7. Patrón fachada	81
5.8. Patrón método factoría	82
5.9. Diagrama de clases de diseño	90
5.10. CU01 - Registro en el sistema	91
5.11. CU02 - Inscripción en el sistema	92
5.12. CU03 - Verificación de locutor	93
7.1. Diagrama de flujo de los front-ends	122

Índice de tablas

2.1. Resultados prueba 1	18
2.2. Resultados prueba 2	18
2.3. Resultados prueba 3	19
2.4. Resultados prueba 4	20
3.1. Roles y responsabilidades	26
3.2. Actividades de la fase de planificación	29
3.3. Actividades de la fase de análisis	30
3.4. Actividades de la fase de diseño	31
3.5. Actividades de la fase de implementación	32
3.6. Actividades de la fase de pruebas (módulos)	33
3.7. Actividades de la fase de pruebas (programa completo)	34
3.8. Actividades finales de documentación	35
3.9. Costes de personal	36
3.10. Riesgo 01 - Cambio del análisis	39
3.11. Riesgo 02 - Reducción de la disponibilidad de un miembro del equipo	40
3.12. Riesgo 03 - Fallo en los equipos	41
3.13. Riesgo 04 - Poca destreza con las herramientas	42
3.14. Riesgo 05 - Estimaciones imprecisas	43
3.15. Riesgo 06 - Problemas con el uso de las herramientas	44
3.16. Fases en UPEDU	47
4.1. Stakeholders del sistema	51
4.2. Necesidades y características	52
4.3. Matriz de trazabilidad	62
7.1. T-ALI01 - Nombre de usuario vacío	103
7.2. T-ALI02 - Nombre de usuario compuesto por caracteres en blanco	103
7.3. T-ALI03 - Creación correcta de usuario	104
7.4. T-ALI04 - Extracción de características con nombre de fichero vacío	104
7.5. T-ALI05 - Extracción de características con fichero inexistente	104
7.6. T-ALI06 - Extracción de características con fichero sin permisos de lectura	105

7.7. T-ALI07 - Extracción de características correcta	105
7.8. T-ALI08 - Normalización de energía	105
7.9. T-ALI09 - Normalización de energía sin fichero de características	106
7.10. T-ALI10 - Detección de energía	106
7.11. T-ALI11 - Detección de energía sin fichero de características normalizadas	106
7.12. T-ALI12 - Normalización de características	107
7.13. T-ALI13 - Normalización de características sin ficheros necesarios	107
7.14. T-ALI14 - Creación del modelo	107
7.15. T-ALI15 - Creación de modelo sin ficheros necesarios	108
7.16. T-ALI16 - Verificación de usuario	108
7.17. T-ALI17 - Verificación de usuario sin modelo	108
7.18. T-ALI18 - Verificación de usuario sin fichero de características	109
7.19. T-ALI19 - Borrado de usuario	109
7.20. Particiones de equivalencia - Registro de usuario	111
7.21. T-API01 - Registro de usuario correcto	112
7.22. T-API02 - Registro de usuario con nombre de usuario vacío	112
7.23. T-API03 - Registro de usuario con nombre de usuario menor de dos caracteres	112
7.24. T-API04 - Registro de usuario con nombre de usuario mayor de seis caracteres	113
7.25. T-API05 - Registro de usuario con nombre de usuario ya registrado	113
7.26. T-API06 - Registro de usuario con nombre de usuario inválido	113
7.27. T-API07 - Registro de usuario con nombre vacío	114
7.28. T-API08 - Registro de usuario con nombre mayor de 30 caracteres	114
7.29. T-API09 - Registro de usuario con nombre inválido	114
7.30. T-API10 - Registro de usuario con apellido vacío	115
7.31. T-API11 - Registro de usuario con apellido mayor de 100 caracteres	115
7.32. T-API12 - Registro de usuario con apellido inválido	115
7.33. T-API13 - Registro de usuario con email vacío	116
7.34. T-API14 - Registro de usuario con email inválido	116
7.35. T-API15 - Registro de usuario con email mayor de 50 caracteres	116
7.36. T-API16 - Registro de usuario con email registrado previamente	117
7.37. Particiones de equivalencia - Inscripción de locutor	117
7.38. T-API17 - Inscripción de locutor	117
7.39. T-API18 - Inscripción de locutor con usuario no registrado	118
7.40. T-API19 - Inscripción de locutor con fichero inválido	118
7.41. Particiones de equivalencia - Verificación de locutor	119
7.42. T-API20 - Verificación de locutor	119
7.43. T-API21 - Verificación de locutor con usuario no inscrito	119
7.44. T-API22 - Verificación de locutor con fichero inválido	120

7.45. Casos de prueba - Flujo en front-ends 121

B.1. Cambios en el documento 137

Capítulo 1.

Introducción

1.1. Descripción del proyecto

El grupo de investigación de la universidad de Valladolid ECA-SIMM (Entornos de Computación Avanzada y Sistemas de Interacción MultiModal), lleva varios años trabajando en el área de la biometría. Fruto de ese trabajo, se desarrolló hace más de diez años un sistema de reconocimiento de voz, y una página web, compatible con PDAs, los dispositivos móviles predominantes en la época, para permitir su uso a cualquier entidad externa de una manera sencilla.

Las limitaciones de las tecnologías existentes en el momento hacen necesaria una actualización completa del sistema, para ajustarlo a los estándares de calidad y las técnicas de ingeniería de software actuales. Entre otras, algunas de las carencias que presenta el sistema anterior son la ausencia de orientación a objetos o de control de errores, y la falta de separación entre las capas de datos, modelo de negocio y vista de la aplicación, separación denominada MVC (Modelo-Vista-Controlador).

El trabajo anterior ha sido utilizado, únicamente, para obtener los requisitos de usuario del nuevo desarrollo tratado en este documento. En ningún caso se ha reutilizado el código o los ejecutables que daban soporte a esta plataforma. Por lo tanto, el proyecto presentado se construye sin ninguna base establecida.

1.1.1. Metodología utilizada

Este proyecto se presenta como un ejercicio completo de ingeniería de software, desde la planificación temporal al despliegue de la solución completa, siguiendo la estructura del proceso unificado, en concreto la propuesta por UPEDU, con las modificaciones pertinentes para adaptarlo a la realización de un proyecto académico con un solo integrante.

Para poder llevar a cabo esta metodología de trabajo se han consultado y adaptado las plantillas que ofrece el proyecto OpenUP.

1.2. Objetivos del proyecto

1.2.1. Objetivo general

Actualizar de manera completa un sistema de reconocimiento biométrico de usuario mediante voz, para ajustarlo a los estándares de calidad y técnicas de ingeniería de software actuales.

Del sistema anterior sólo se van a reutilizar los requisitos, de manera que el resto será completamente nuevo, para poder ser actualizado a las nuevas herramientas de programación web y plataformas móviles.

1.2.2. Objetivos específicos

Los objetivos específicos, que, como su propio nombre indica, se encuentran mucho más cercanos a las competencias específicas que los generales, se describen a continuación:

- El sistema permitirá verificar la identidad del usuario mediante voz tanto vía web como mediante dispositivos móviles Android.
- Se analizará el sistema anterior para extraer los requisitos del nuevo proyecto.
- Para el desarrollo del nuevo sistema se utilizarán técnicas de programación modernas y buenas prácticas reconocidas, para mejorar la calidad y el desempeño del sistema anterior. En concreto, el uso del paradigma de orientación a objetos, control de excepciones, generación de documentación interna y patrón modelo vista controlador.
- El sistema final deberá ser mantenible y reutilizable en futuros proyectos del grupo de investigación reconocido de la UVA ECA-SIMM.
- Garantizar la solidez del sistema, mediante un diseño de pruebas extenso, que incluya el desarrollo de pruebas unitarias y de integración.
- Aunque no es objetivo del presente proyecto, se procurará que el sistema de reconocimiento funcione lo mejor posible.
- Tanto para el desarrollo como para la implementación del sistema se usarán herramientas y software gratuito.
- El sistema final deberá ser fácil de usar.

1.3. Estructura del documento

El anterior objetivo general se concreta en los siguientes objetivos específicos:

- **Introducción:** Es el capítulo en el que nos encontramos actualmente, se describe de forma general el proyecto, los objetivos a lograr, la metodología utilizada y los contenidos de esta memoria.
- **Estado del arte:** Se realiza un estudio del estado del arte en lo referente al reconocimiento de locutor, estudiando los sistemas disponibles para dar soporte a nuestro proyecto y probando una configuración lo más optimizada posible.
- **Plan de desarrollo:** Se detalla la división en tareas y la secuenciación y temporización de estas. Se definen los riesgos, su posible impacto y la probabilidad de su aparición.
- **Análisis:** Partiendo del proyecto que se pretende actualizar, se obtienen los requisitos de usuario, y a partir de ellos los requisitos funcionales y no funcionales. Se incluyen los diagramas de análisis necesarios para dar una primera visión del sistema.
- **Diseño:** Detallamos la arquitectura concreta en la que el proyecto se desplegará, creando las vistas lógicas y físicas del sistema. Se establecen los criterios de calidad de diseño, describiendo también las estrategias seguidas para conseguir alcanzarlos. Se traducen los casos de uso y diagramas de análisis a la visión detallada de diseño.
- **Implementación:** Se describen las decisiones adoptadas a bajo nivel, como el software elegido y las herramientas utilizadas para el desarrollo, despliegue y testing del proyecto.
- **Pruebas:** Queda descrito el diseño de las pruebas, con las particiones de equivalencia y casos de prueba necesarios para cada componente desarrollado.
- **Control, seguimiento y plan de cierre:** En este capítulo se reflejan los resultados del proceso de realización del proyecto, comparando la previsión realizada al inicio con los resultados obtenidos finalmente, teniendo en cuenta los riesgos sufridos y los objetivos conseguidos.
- **Conclusiones y trabajo futuro:** Conclusiones generales de la realización del trabajo fin de grado, y líneas en las que continuar el trabajo en un futuro.
- **Bibliografía:** Listado de las referencias bibliográficas utilizadas.

Capítulo 2.

Estado del arte

2.1. Introducción

En este capítulo se aborda la investigación de la situación actual en el campo del reconocimiento de locutor. Se estudiarán las herramientas disponibles para incluir en el proyecto y se realizarán pruebas de la herramienta escogida para obtener la configuración más óptima de cara al éxito final del desarrollo.

2.2. Reconocimiento de locutor

El reconocimiento de locutor es una de las disciplinas enmarcadas en la biometría. Engloba la verificación de locutor, consistente en comprobar que una persona es quien dice ser mediante el estudio de su voz, y la identificación de locutor, en la que el sistema decide quién es el locutor dentro de un grupo dado. Podemos afirmar, por tanto, que el resultado de la verificación es una respuesta verdadero/falso y el de la identificación es un usuario.

Dentro del reconocimiento de locutor, independientemente de si se trata de verificación o identificación, cabe distinguir el análisis independiente o dependiente de texto. Cuando se realiza un análisis independiente de texto, el sistema debe conseguir reconocer la identidad del locutor sin importar lo que este diga en sus grabaciones. En el caso del análisis dependiente de texto, la grabación de la voz del locutor deberá estar formada por la pronunciación de una serie de palabras o letras prefijadas. [Jr.97]

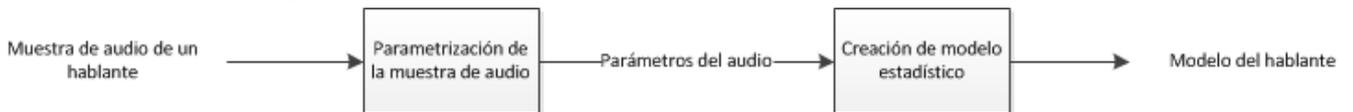
Una vez definidas las distintas variantes de reconocimiento de locutor, podemos enmarcar el trabajo que se desarrollará en este proyecto en el de la **Verificación** de locutor **independiente** de texto. Será esta modalidad la que estudiaremos durante este capítulo, no sin perder de vista otras que puedan resultar interesantes.

2.3. Verificación de locutor independiente de texto

La verificación de locutor se compone de dos fases, como la de cualquier otro reconocimiento biométrico, la fase de entrenamiento y la de operación.

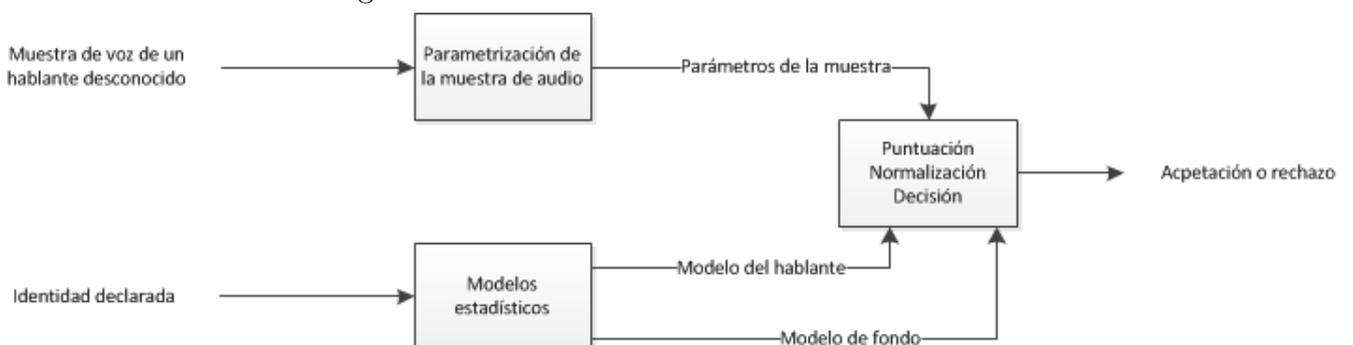
En la fase de entrenamiento, primero se extraen las características únicas de la voz tratada, obteniendo así una representación de esta apta para su tratamiento matemático. De esta representación, a continuación, se genera un modelo estadístico, único para cada hablante. Este proceso se refleja en la figura 2.1.

Figura 2.1.: Fase de entrenamiento en verificación de locutor



En la fase de operación, se tienen como entradas una muestra de audio y la identidad del hablante al que, supuestamente, pertenece. En primer lugar, al igual que durante el entrenamiento, se extraen las características del audio. A continuación se recuperan el modelo del hablante declarado y el modelo de fondo del sistema, que se habrán calculado anteriormente, durante la fase de entrenamiento. Finalmente, se halla el valor de similitud de las características del audio comparándolo con el modelo, utilizando el modelo de fondo, se normalizan los resultados y se da una respuesta sobre la validez o no de la muestra de audio para el usuario declarado [BBF⁺04]. Podemos ver el proceso en la figura 2.2.

Figura 2.2.: Fase de test en verificación de locutor



2.3.1. Parametrización de la muestra de audio

La parametrización consiste en la transformación del audio en una serie de vectores de características. Esta transformación nos permite obtener una representación con la que es más sencillo trabajar, al ser más compacta y proveer mayor facilidad de cálculo que un fichero de audio. Una de las características más utilizadas en el reconocimiento de locutor son los coeficientes cepstrales. Estas características representan, de alguna manera, la forma del tracto vocal, siendo este particular de cada hablante.

Existen dos técnicas para obtener esta representación cepstral, la basada en “banco de filtros” y la basada en “LPC” (*Linear Predictive Coefficients*, coeficientes del tracto vocal modelado como filtro digital).

La representación cepstral basada en “filterbank” (representada en la figura 2.3) tiene, a su vez, varias fases, que, de forma resumida, consisten en lo siguiente:

Figura 2.3.: Parametrización basada en filterbank



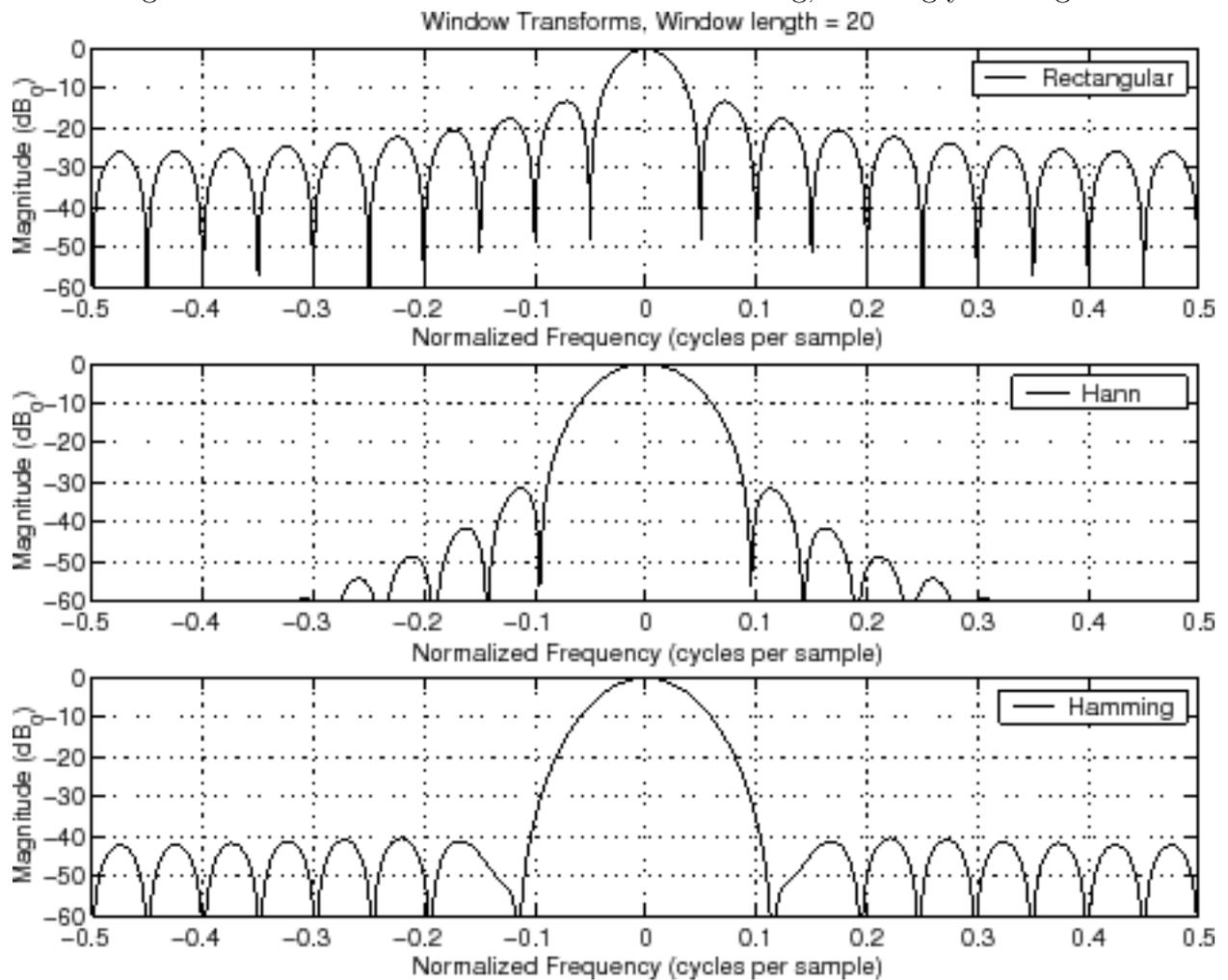
- **Pre-énfasis:** Se aplica un filtro para dar mayor potencia a las frecuencias altas del audio, que suelen ser mitigadas durante el proceso de verificación de locutor. El filtro aplicado es el siguiente:

$$x_p(t) = x(t) - a \cdot x(t - 1) \quad (2.1)$$

El valor de a se encuentra normalmente en el rango $[0.95, 0.98]$.

- **Ventanizado:** El ventanizado consiste en dividir la señal en fragmentos más pequeños, comenzando por el inicio hasta que se llega al fin de esta. Los valores clave para este paso son el tamaño de la ventana (entre 20 y 30 milisegundos, habitualmente) y el del solapamiento entre ventanas (normalmente 10 milisegundos). Una vez establecidos estos valores se determinará qué tipo de ventana utilizar, siendo las más habituales la de Hamming o la de Hanning. Ambas son más efectivas a la hora de aplicar los siguientes pasos que la opción restante, la rectangular, ya que genera unos bordes de la ventana más suaves. Observamos en la figura 2.4 las diferencias entre ellas.
- **Transformada rápida de Fourier:** Normalmente nombrada como FFT (*Fast Fourier Transform*), es un algoritmo que permite computar de forma más eficiente la DFT (Transformada discreta de Fourier), función matemática que extrae los componentes frecuenciales

Figura 2.4.: Transformaciones de ventana: Hamming, Hanning y rectangular



de una señal periódica. Una vez calculadas las ventanas, se aplica a cada una de ellas cualquier algoritmo FFT de los conocidos. Debemos establecer el número de puntos de la función que indican el número de componentes frecuenciales que se extraerán de la señal, siendo este número fijado normalmente en 512. Del espectro de 512 puntos obtenido, y dado que la FFT es simétrica, se mantienen sólo los 256 primeros.

- **Envolvente espectral:** El espectro que tenemos en este momento tendrá grandes fluctuaciones, que no nos resultan interesantes. Debemos obtener la envolvente espectral multiplicando dicho espectro por un filterbank.
- **20 · Log:** Calculando el logaritmo de la envolvente espectral y multiplicando cada coeficiente por 20 convertimos las unidades a dB.
- **Transformación cepstral:** Por último, se aplica una transformación de coseno discreta para convertir los vectores espectrales a cepstrales.

La representación basada en LPC tiene también las fases opcionales de pre-énfasis y ventanizado, pero posteriormente usa un filtro AR, para modelar el tracto vocal, cuyos parámetros se calculan para cada una de las ventanas. Se obtienen así una serie de vectores LPC que podrán ser transformados en cepstrales. La figura 2.5 describe este proceso.

Figura 2.5.: Parametrización basada en LPC



Una vez obtenidas las cepstrales es habitual realizar la técnica conocida como sustracción de la media del cepstrum (CMS, *Cepstral Mean Substraction*) o centrado. Así se atenúa el efecto del “canal”, es decir, de los componentes frecuenciales introducidos en la señal de voz por los medios técnicos usados para su captura (micrófono, teléfono, etc.). En este proceso, además, habitualmente, se normaliza la varianza de cada componente a uno, es decir, se realiza una reducción.

En muchas ocasiones, sobre todo cuando se obtienen las cepstrales en un proceso orientado al reconocimiento de locutor, tras los posibles procesos de centrado y reducción se añade información dinámica de cada vector, añadiendo la primera y segunda derivadas que generan, además de la energía.

Por último, se deben descartar los vectores que menos información proporcionen, es decir, los que no correspondan a fragmentos de voz. Esta operación se realiza calculando un modelo bi-Gaussiano de la distribución de vectores. El vector con media más baja corresponderá al silencio,

y el que tenga una media más alta a conversación. El resto de vectores se aceptan o descartan dependiendo de la similitud de su media con las dos anteriores.

2.3.2. Modelado estadístico

La decisión de si una muestra de audio Y corresponde a un hablante S , se puede determinar como un cociente de probabilidad. Para ello debemos plantear dos hipótesis:

H_0 : La muestra Y pertenece a S

H_1 : La muestra Y *no* pertenece a S

El cociente de probabilidad para decidir qué hipótesis es la correcta sería, por tanto:

$$\frac{p(Y|H_0)}{p(Y|H_1)} = \begin{cases} > \theta & \text{aceptada } H_0 \\ < \theta & \text{aceptada } H_1 \end{cases} \quad (2.2)$$

El coeficiente Y corresponde a el conjunto de vectores que hemos obtenido durante la parametrización del audio. Para calcular la probabilidad de que la hipótesis H_0 sea cierta, usaremos el modelo del usuario al que, supuestamente, Y pertenece.

Sin embargo, calcular la probabilidad de que Y esté en H_1 no se puede hacer de una forma tan sencilla, porque no tenemos un modelo “contrario” al del hablante, la cantidad de modelos posibles sería infinita. La práctica más habitual, y que utilizaremos en este proyecto, es la de generar un único modelo de fondo, combinando un conjunto de hablantes que cubran, si es posible, todo el espectro de tipos de hablantes que van a utilizar el sistema. Este modelo se denomina, en la mayoría de la literatura disponible, UBM (*Universal Background Model*).

El modelo estadístico que utilizaremos para estos cálculos es la de Modelos de Mezclas Gaussianas (GMM, *Gaussian Mixture Models*). No es propósito de este proyecto entrar en detalles de la generación del GMM, sólomente mencionaremos que el modelo de fondo es un gran GMM, que engloba las características de un conjunto de voces, y que nos servirá para comprobar la hipótesis H_1 del cociente de probabilidad de la verificación de locutor.

2.3.3. Normalización

Tras generar el modelo de fondo GMM/UBM y los modelos de entrenamiento de los hablantes a tratar, podemos calcular el cociente de probabilidad y, a partir de él, tomar una decisión sobre si la muestra de voz se corresponde con el hablante declarado o no.

Esto se realiza estableciendo un umbral de decisión. Si el resultado del cociente de probabilidad es mayor que el umbral, aceptaremos la hipótesis de que la muestra de voz pertenece al hablante, y lo contrario si este cociente es menor.

Es en el establecimiento del valor de este umbral donde debemos ser especialmente cuidadosos, y es que hay una gran cantidad de factores que pueden hacer que la variabilidad del cociente de probabilidad obtenido en nuestros experimentos sea muy alta.

Para poder establecer de una forma algo más sencilla el umbral de decisión, se han desarrollado diferentes técnicas de normalización, que intentan reducir la variabilidad en los resultados de las verificaciones de locutor. La fórmula básica que guía todas ellas es la siguiente:

$$\tilde{L}_\lambda(X) = \frac{L_\lambda(X) - \mu_\lambda}{\sigma_\lambda} \quad (2.3)$$

Donde $L_\lambda(X)$ es el resultado de la verificación de locutor (ecuación 2.2) para una muestra de audio X contra un modelo λ , siendo $\tilde{L}_\lambda(X)$ el resultado normalizado. Los factores μ_λ y σ_λ son los parámetros concretos para el hablante λ , y son el objeto del estudio de la normalización, ya que son los que deben ser estimados para producir los mejores resultados posibles.

Hay varias técnicas de normalización basadas en esta premisa, las más conocidas son ZNorm, HNorm, TNorm, CNorm y las combinaciones de estas (se puede encontrar una descripción de estas técnicas en [JPDCD09] y [LJ09]). Durante las pruebas, utilizaremos varias de ellas, para elegir la que mejor convenga a nuestro sistema.

2.3.4. Evaluación

Para evaluar el rendimiento de un sistema de verificación de locutor se tienen que tener en cuenta los dos tipos de errores posibles, o bien aceptar como válida la muestra de audio de un impostor (falso positivo), o no aceptar como válida la muestra de audio de un usuario que sí que pertenece a este (falso negativo).

Estos dos tipos de errores están directamente influenciados por el umbral θ elegido. En caso de ser este muy bajo, habrá muchos falsos positivos, y si se establece un valor alto, se darán muchos falsos negativos. La clave para optimizar el rendimiento de un sistema de verificación de locutor

es encontrar un valor para este umbral que garantice un equilibrio entre ambos tipos de errores, este valor es denominado el “punto de operación del sistema”.

El cálculo del punto de operación se realiza mediante pruebas experimentales sobre un conjunto de muestras de hablantes, generando el modelo con una parte de ellas y realizando las pruebas de verificación de identidad con el resto, de manera que las pruebas nunca se hagan sobre hablantes incluidos en el modelo de fondo.

Una vez obtenidos los datos del conjunto de hablantes elegidos se suele representar la curva DET de los resultados, que enfrenta de forma gráfica, utilizando una escala normalizada, la cantidad de falsos positivos con la de falsos negativos. El rendimiento del sistema será mejor cuanto más se acerque la curva de dicha gráfica al origen de los ejes. Utilizando esta curva también podemos obtener el valor TEE, tasa de equierror (*EER*, *equal error rate*), que es el punto en el que la cantidad de errores de ambos tipos es igual (en términos matemáticos, la intersección de la curva DET con la bisectriz de los ejes).

Ambos valores serán estudiados en la sección 2.5, en la que probaremos el software elegido para el sistema, buscando la configuración más óptima.

2.4. Software para el reconocimiento de locutor

Necesitamos varios tipos de software para realizar el proceso completo de reconocimiento de locutor. Los detallamos a continuación, indicando la opción elegida para nuestro proyecto en cada una de las fases de proceso del audio.

2.4.1. Tratamiento de audio

En primer lugar, un software destinado al tratamiento de audio. A la hora de obtener los parámetros de una muestra de audio vamos a utilizar software libre disponible. Uno de los problemas de esta opción es que cada paquete acepta el audio en un determinado formato. El que vamos a utilizar usa, por ejemplo, formato PCM (Pulse Code Modulation) o lo que es lo mismo, un fichero en formato WAV pero sin cabecera. Lo normal, al grabar audio, es que este se grabe en formato comprimido de alguna manera o que, al menos, se le añada una cabecera, por lo que deberemos convertir esos ficheros al formato adecuado al software de extracción de características a usar. Para esta tarea utilizaremos **sox**, disponible para cualquier sistema operativo y que cumple todas las funciones requeridas.

2.4.2. Extracción de características

Como se ha comentado, usaremos herramientas libres para la extracción de características. Más concretamente, vamos a utilizar una de las más usadas, las SPRO Tools¹, que proporcionan dos ejecutables para realizar la extracción de las cepstrales: **sfbcep** utilizando filterbank y **slpcep** utilizando LPC. Utilizaremos el primero.

2.4.3. Verificación de locutor

Por último, debemos realizar el reconocimiento de locutor propiamente dicho. Existen varios paquetes de autenticación biométrica disponibles en la actualidad. Nos centraremos en dar una breve descripción de los que podemos usar en este proyecto, es decir, los que son gratuitos y de código abierto, que además se puedan incorporar como librerías o interfaces a nuestro sistema.

- **Sistema legado del GIR ECA-SIMM** Este es el sistema que se utilizó en el proyecto al que da origen el que trata este documento. Tiene ventajas evidentes como el gran conocimiento que de él dispone el tutor del proyecto, haciendo que su ayuda en la realización

¹Dirección web de las SPRO Tools: <http://www.irisa.fr/metiss/guig/spro/>

del proyecto pueda ser muy precisa, y que ha sido un desarrollo satisfactorio, que ya se ha probado con resultados positivos por parte de los actores implicados en su uso y creación.

Conocidas las ventajas, este sistema es una opción sólida para acompañar a este proyecto, sin embargo las pruebas que se han hecho con él demuestran que es harto complicado compilarlo y ejecutarlo en un sistema moderno, al depender de librerías difíciles de encontrar y estar pensado para arquitecturas más antiguas.

- **SPEAR** Presenta un sistema de código abierto para el reconocimiento de locutor basado en BOB. Está escrito en Python, lo que conlleva la gran ventaja de poder instalarse sobre cualquier tipo de sistema operativo y, por otra parte, la desventaja de no ser tan eficiente como sus posibles contrapartidas en lenguajes de menos alto nivel, como C ó C++. [KESM14]

Una exploración de la información disponible sobre este sistema nos da muy pocos resultados, y es que la relativa novedad de su desarrollo (el paper que dio a conocer este software es del año 2014), hacen que sea una apuesta arriesgada, por la probable poca madurez con la que aún cuenta su desarrollo y los errores no resueltos con los que podamos tener que lidiar.

- **BECARS** Este software se compone de una librería escrita en C. En su creación colaboran la universidad de Universidad de Balamand en el Líbano y la Escuela nacional superior de telecomunicaciones de París. Este software ha sido ampliamente utilizado en ámbitos académicos y dispone de gran cantidad de documentación. [BMM⁺04]
- **Alize** Es una plataforma de código abierto (bajo la licencia LGPL) para la autenticación biométrica escrita en C++. Su desarrollo corre a cargo de la Universidad de Avignon, y está compuesta por una librería de bajo nivel (Alize) y un toolkit con herramientas de alto nivel para realizar las acciones propias del reconocimiento de locutor (LIA_RAL). [LBF⁺13]

Destaca la cantidad de literatura disponible disponible para documentar su uso que, sin ser realmente extensa, destaca en comparación con los otros sistemas. Tiene como ventaja sobre BECARS el hecho de que puede generar y manejar i-vectores, una técnica exitosa en la verificación de locutor dependiente de texto.

La elección que realizaremos en este caso será la de **Alize**. Principalmente por contar con mayor documentación que el resto de las opciones, pero también por ser su última versión relativamente reciente pero contar con varios años de desarrollo y por las características adicionales que nos proporciona, útiles para futuras ampliaciones del sistema.

2.5. Pruebas con Alize

La realización de las pruebas con Alize, el software elegido, tiene como objetivo generar un modelo de fondo adecuado y encontrar la configuración más óptima para el buen rendimiento de nuestro sistema, así como elegir el umbral más cercano al punto de operación de este.

Para ello, contamos con un corpus de prueba formado por voces en español recogidas con micrófono, por lo que las muestras de audio serán de la mayor similitud posible con respecto a las que se recogerán mediante el uso del sistema implementado.

2.5.1. Datos del corpus

El corpus que se va a utilizar cuenta con muestras de audio de 132 hablantes distintos. Cada uno de ellos ha realizado cuatro sesiones de grabación en distintos momentos, y en cada una de esas sesiones ha grabado un máximo de once muestras de audio que pueden ser de varios tipos:

- Frase
- Clave numérica auténtica
- Clave numérica falsa

Las claves numéricas se suelen utilizar en un sistema de reconocimiento de locutor dependiente de texto, sin embargo, en nuestro caso, simplemente sirven para añadir datos de su voz al modelo. La diferencia entre la clave numérica verdadera y las falsas es que la primera es siempre la misma y se graba varias veces por sesión, mientras que las segundas son distintas entre ellas y, cada una, se graba únicamente una vez por sesión.

Cada una de estas once muestras (como máximo) de audio por sesión y hablante hace que tengamos un máximo de 44 ficheros de audio por usuario, todos ellos de muy corta duración, por lo que para poder realizar un estudio de cada hablante será necesario unir varios ficheros, creando audios de mayor duración, tanto para la generación del modelo como para las pruebas. La cantidad de ficheros a unir y el tipo de estos será uno de los parámetros de cada uno de los experimentos que se hagan.

2.5.2. Procedimientos involucrados en las pruebas

Alize y LIA_RAL generan varios ejecutables como resultado de su compilación. Cada uno de estos ejecutables tiene su función dentro del proceso de verificación de locutor, y hay ejecutables específicos para el tratamiento de i-vectores o de modelos GMM-UBM. La cadena de procesos que se seguirá para probar el software es la siguiente:

- **sfbcep**: El primer paso para la verificación de locutor no está incluido dentro del framework de Alize, sino dentro de las SPRO Tools. sfbcep, descrito en la sección 2.4.2, extrae las características del audio.
- **NormFeat (para normalización de energía)**: Se utiliza el ejecutable NormFeat para normalizar la energía, proceso equivalente a eliminar los vectores que corresponden a silencio dentro de la muestra.
- **EnergyDetector**: Este ejecutable se encuentra de crear un fichero de texto, en el que se indica las partes en las que el sonido escuchado es voz del hablante.
- **NormFeat (para normalización de características)**: Se normaliza el cepstrum para eliminar los vectores menos representativos.
- **TrainWorld**: Genera el modelo de fondo a partir de un conjunto de hablantes, representados cada uno de ellos por su fichero de parámetros.
- **TrainTarget**: Genera un modelo para cada hablante. Se le proporciona una lista de los hablantes de test y obtiene el modelo a partir de sus características.
- **ComputeTest**: Compara un fichero de parámetros con el modelo de un hablante, de tal forma que da una puntuación de la certeza que se tiene en afirmar que el fichero de audio, representado por los parámetros, pertenece al hablante del que se extrajo el modelo. También, dependiendo de las opciones, este ejecutable normalizará el resultado con el procedimiento que se escoja.

2.5.3. Parámetros de las pruebas

Para llevar a cabo todo el proceso de reconocimiento de locutor se deben aportar al sistema varios parámetros de entrada en forma de opciones de configuración. Distinguiremos parámetros fijos, los que se utilizarán en todos los casos de prueba y que se definirán previamente a estas, de los parámetros variables, que son los que serán modificados para obtener los mejores resultados posibles.

Parámetros fijos

El primer conjunto de parámetros fijos es el de los relacionados con el fichero PCM creado. Este tendrá siempre un solo canal de audio, 16 bits por muestra y una frecuencia de muestreo de 8000Hz.

Dentro de los parámetros fijos están los relacionados con la obtención de los parámetros de audio, que son estándar de facto en el dominio del reconocimiento de locutor. Se trata de los siguientes

- Pre-énfasis: 0.97
- Tamaño de ventana: 20ms
- Tamaño de solapamiento entre ventanas: 10ms
- Función de ventana: Hamming
- Número de cepstrales: 19

Añadiremos siempre, además, la cepstral con la energía y las derivadas primera y segundas de cada cepstral, resultando en un vector de características por ventana con 60 componentes.

Para el centrado (conocido en Alize como detección de energía) aplicaremos la técnica habitual de sustracción de la media, y la normalización de las características se realizará sobre una distribución Gaussiana $\mathcal{N}(0, 1)$.

Parámetros variables

Los parámetros variables son, en primer lugar, la cantidad de muestras de audio de cada sesión y la cantidad de sesiones que utilicemos para cada hablante al generar el modelo de fondo o los casos de prueba.

En segundo lugar, se probará con distintos números de Gaussianas y de iteraciones al generar el modelo de fondo.

2.5.4. Resultados de las pruebas

Las pruebas van a arrojar cuatro resultados:

- Umbral de decisión de la Tasa de Equierror
- Tasa de equierror
- Porcentaje de falsos negativos
- Porcentaje de falsos positivos

Este conjunto de resultados se obtendrá utilizando ComputeTest tanto sin normalización como con normalizaciones TNorm, ZNorm y ZTNorm.

Prueba 1 - Configuraciones sin optimizar

Número de hablantes del modelo: 72

Número de hablantes para test: 30

Número de hablantes usados para la normalización TNorm: 15

Número de hablantes usados para la normalización ZNorm: 15

Número de sesiones de grabación para los hablantes del modelo de fondo: 1

Número de muestras: 1

Número de sesiones de grabación para entrenar el modelo de cada hablante: 1

Número de muestras de voz para entrenar el modelo de cada hablante de test: 4

Número de sesiones de grabación para los hablantes usados para ZNorm: 1

Número de muestras: 4

Número de sesiones de grabación para los hablantes usados para TNorm: 1

Número de muestras: 4

Gaussianas en el modelo de fondo: 32

Tabla 2.1.: Resultados prueba 1

	Umbral medio	TEE (%)	Fn (%)	Fp (%)
Sin norm	0.15	4.63	4.58	4.67
ZNorm	1.91	6.49	6.45	6.52
TNorm	1.51	9.62	9.58	9.66
ZTNorm	1.47	8.55	8.54	8.56

Prueba 2 - Aumento de muestras para el modelo

Como primer intento de mejora, aumentamos el número de muestras de cada hablante utilizadas para generar el modelo de fondo de 1 a 3. De forma que tenemos la configuración siguiente:

Número de hablantes del modelo: 72

Número de hablantes para test: 30

Número de hablantes usados para la normalización TNorm: 15

Número de hablantes usados para la normalización ZNorm: 15

Número de sesiones de grabación para los hablantes del modelo de fondo: 1

Número de muestras: 3

Número de sesiones de grabación para entrenar el modelo de cada hablante: 1

Número de muestras de voz para entrenar el modelo de cada hablante de test: 4

Número de sesiones de grabación para los hablantes usados para ZNorm: 1

Número de muestras: 4

Número de sesiones de grabación para los hablantes usados para TNorm: 1

Número de muestras: 4

Gaussianas en el modelo de fondo: 32

Tabla 2.2.: Resultados prueba 2

	Umbral medio	TEE (%)	Fn (%)	Fp (%)
Sin norm	0.14	4.79	4.79	4.79
ZNorm	1.91	5.93	5.83	5.91
TNorm	1.53	9.00	8.95	9.04
ZTNorm	1.67	7.71	8.54	8.56

Prueba 3 - Aumento de gaussianas en el modelo

La prueba anterior constata que el aumento de muestras utilizadas para generar el modelo no mejora los resultados. Enfocamos, por tanto, las pruebas en otra dirección, aumentaremos el número de Gaussianas del modelo de fondo, para obtener una representación de este más detallada.

Número de hablantes del modelo: 72

Número de hablantes para test: 30

Número de hablantes usados para la normalización TNorm: 15

Número de hablantes usados para la normalización ZNorm: 15

Número de sesiones de grabación para los hablantes del modelo de fondo: 1

Número de muestras: 3

Número de sesiones de grabación para entrenar el modelo de cada hablante: 1

Número de muestras de voz para entrenar el modelo de cada hablante de test: 4

Número de sesiones de grabación para los hablantes usados para ZNorm: 1

Número de muestras: 4

Número de sesiones de grabación para los hablantes usados para TNorm: 1

Número de muestras: 4

Gaussianas en el modelo de fondo: 128

Tabla 2.3.: Resultados prueba 3

	Umbral medio	TEE (%)	Fn (%)	Fp (%)
Sin norm	0.11	5.39	5.41	5.35
ZNorm	1.91	6.71	6.67	6.75
TNorm	1.51	10.01	10.00	10.02
ZTNorm	1.51	9.41	9.38	9.45

Prueba 4 - Aumento de usuarios en el modelo

Tampoco apreciamos una mejora significativa, pero el aumento del número de Gaussianas nos permite aumentar el número de hablantes utilizados para generar el modelo. Aplicaremos este cambio a la configuración de las pruebas.

Número de hablantes del modelo: 102

Número de hablantes para test: 15

Número de hablantes usados para la normalización TNorm: 15

Número de hablantes usados para la normalización ZNorm: 15

Número de sesiones de grabación para los hablantes del modelo de fondo: 1

Número de muestras: 3

Número de sesiones de grabación para entrenar el modelo de cada hablante: 1

Número de muestras de voz para entrenar el modelo de cada hablante de test: 4

Número de sesiones de grabación para los hablantes usados para ZNorm: 1

Número de muestras: 4

Número de sesiones de grabación para los hablantes usados para TNorm: 1

Número de muestras: 4

Gaussianas en el modelo de fondo: 128

Tabla 2.4.: Resultados prueba 4

	Umbral medio	TEE (%)	Fn (%)	Fp (%)
Sin norm	0.16	3.08	3.13	3.04
ZNorm	2.58	4.46	4.38	4.45
TNorm	1.94	9.13	9.16	9.08
ZTNorm	1.38	9.33	9.38	9.28

En esta ocasión sí que mejoramos sustancialmente los resultados.

2.5.5. Conclusiones

La conclusión principal es que, cuanto mayor es el número de hablantes utilizados para el modelo, y cuantas más gaussianas incluyamos en el modelo de fondo, menor tasa de error medio se obtiene. Sin embargo, llega un punto en el que, por más hablantes que incluyamos, no obtendremos mejoras sustanciales, y la cantidad de gaussianas no puede ser aumentada si no se añaden nuevas muestras al corpus. En estas pruebas, hemos llegado a ese punto con las muestras de las que disponemos.

Será, por tanto, el modelo de fondo generado en la prueba 4 el que será utilizado en la aplicación final, sin embargo, no debemos utilizar el umbral medio generado, ya que las voces grabadas mediante la web y la app se habrán obtenido en unas condiciones totalmente distintas a las que tienen las voces que forman el corpus. El umbral se ajustará mediante pruebas experimentales una vez que esté desarrollado el sistema.

Capítulo 3.

Plan de desarrollo

3.1. Introducción

La creación de herramientas web para el reconocimiento de locutor no es una disciplina nueva. De hecho, este proyecto se basa en un proyecto anterior, pertenecientes ambos al mismo grupo de investigación: ECA-SIMM (Entornos de Computación Avanzada y Sistemas de Interacción MultiModal).

Este proyecto pretende conseguir, partiendo de la base lograda por el ECA-SIMM, un doble objetivo.

Por una parte, la actualización de la página web BioVoiceWeb utilizando técnicas de la ingeniería de software actuales, que no estaban disponibles o no fueron utilizadas en el momento de la realización del anterior proyecto, como la aplicación del patrón arquitectónico Modelo-Vista-Controlador, orientación a objetos, control de excepciones o patrones software.

Por otra parte, la creación de una aplicación móvil para el sistema operativo Android que permita, de igual forma, realizar las funciones de verificación de locutor.

3.2. Vista general del proyecto

Se presenta en este capítulo toda la información relevante para la elaboración del plan de desarrollo del proyecto, guiado por la plantilla IEEE 1058-1998 [dIEyE98].

3.2.1. Propósitos, alcance y objetivos

La realización de este proyecto tiene como principal objetivo la elaboración del Trabajo de Fin de Grado “Actualización de una aplicación web y móvil para el reconocimiento de usuario por voz”, tal y como ha sido descrito en la Sección 3.1 Introducción.

Junto con ello, se pretenden alcanzar además los objetivos citados a continuación:

- Registro de usuarios en el sistema.
- Inscripción biométrica de usuarios, mediante la grabación de una muestra de voz y posterior creación y almacenamiento del modelo asociado.
- Verificación de locutor, mediante la extracción del modelo de una muestra de voz y la comparación de este con el modelo del usuario inscrito que se haya indicado.

3.2.2. Supuestos y restricciones

Se listan a continuación las restricciones del sistema, tanto funcionales como técnicas, las cuales, a medida que se desarrolle el proyecto irán aumentando:

- El software utilizado para extraer las características del audio y comparar los modelos generados será un software ya desarrollado por terceros. Este proyecto se centra en proveer a los usuarios de interfaces web y móviles para utilizarlo fácilmente.
- Cuestiones relativas al óptimo funcionamiento o la configuración del sistema de verificación de voz utilizado quedan fuera del alcance de este proyecto, siendo parte del “Estado del arte” en esta disciplina.
- No estará permitida la modificación o borrado de los datos cada usuario desde las aplicaciones habilitadas para el reconocimiento de voz.

3.2.3. Entregables del proyecto

Los diferentes entregables del proyecto, descritos a continuación, se desarrollan a lo largo de la lista de Actividades presentada en la Sección 3.4.2 Actividades donde, habitualmente, coinciden con los hitos marcados al final de cada fase con sus respectivas fechas de entrega.

- **Plan de desarrollo del proyecto:** Capítulo actual, donde, como su propio nombre indica, se detallará el plan para conseguir la finalización con éxito del proyecto.
- **Fichero de planificación:** Documento creado con la herramienta Microsoft Project que contiene la planificación a seguir de manera detallada. Indicando actividades, planificación temporal, recursos y descripciones.
- **Estado del arte:** Descripción de la base teórica de las tecnologías utilizadas para realizar el reconocimiento de voz.
- **Análisis:** Capítulo 2. Engloba la identificación de los requisitos de la aplicación y la realización de los casos de uso con sus respectivos diagramas desde el punto de vista del análisis.
- **Diseño:** Descripción y diseño de la arquitectura a seguir en la construcción de la aplicación y realización de los casos de uso con sus respectivos diagramas desde el punto de vista del diseño. Le encontramos en el capítulo 3.
- **Código fuente:** Código fuente de los diferentes módulos que formarán la aplicación, tanto el código de servidor como la página web y la aplicación Android.
- **Aplicación Android:** Código fuente compilado de la aplicación Android.
- **Casos de pruebas:** Resultados de las pruebas de caja blanca y caja negra realizadas tanto sobre los módulos, como sobre la aplicación completa. Para ello se seguirá el plan de desarrollo creado anteriormente para cada tipo de prueba.
- **Manual de instalación:** Instrucciones para poner en funcionamiento la aplicación. Este manual se encuentra en los anexos del presente documento.
- **Manual de usuario:** Documento de ayuda para el usuario. Al igual que el caso anterior, aparece en los anexos.

Debido al método utilizado para llevar a cabo el proyecto, explicado detalladamente en la puesta en marcha del plan, apartado 3.5.1, todos estos entregables podrán sufrir modificaciones a lo largo de su elaboración siguiendo la política de cambios correspondiente.

3.2.4. Resumen de la programación y el presupuesto

Tanto la programación temporal del proyecto, como el presupuesto, se presentarán en los apartados 3.4.1 y 3.4.2 respectivamente.

3.2.5. Evolución del plan

Hay dos sujetos implicados en la realización de este proyecto. Por una parte el alumno y por otra el tutor, siendo el tutor una figura de consulta y apoyo al alumno, y no un actor del proyecto que produzca entregables.

Será, por tanto, el alumno, el encargado de llevar a cabo el proyecto, bajo la supervisión y ayuda del tutor. Dicho proyecto estará formado por una serie de actividades presentadas en el apartado 3.4.2, las cuales serán “realizadas” por los roles descritos en el apartado 3.3.3. No obstante, todos esos roles estarán representados por el alumno, y solo se describen con propósitos académicos.

Las reuniones con el tutor de proyecto se realizarán sin una periodicidad concreta, sino bajo petición de cualquiera de los dos interesados. En todo caso, estas reuniones coincidirán, dentro de lo posible, con los hitos en los que se haya generado la mayor parte de uno o varios entregables.

3.3. Organización del Proyecto

3.3.1. Interfaces externas

Como interfaces externas nos encontramos al grupo de investigación de la Universidad de Valladolid que va a utilizar este proyecto, así como al responsable académico del mismo, de forma más directa.

También se considerará proveedor, tanto al software que realice las operaciones biométricas de verificación de locutor, como a los frameworks o librerías que sea necesario utilizar en el desarrollo. Los diferentes proveedores disponibles serán evaluados y elegidos en la fase de diseño del proyecto.

3.3.2. Estructura interna

Debido a las particularidades propias de este Trabajo de Fin de Grado, realizado por un solo alumno, la estructura interna de la organización del proyecto se limita al alumno y al tutor.

3.3.3. Roles y responsabilidades

Se listan en la siguiente tabla las diferentes responsabilidades de los roles necesarios para llevar a cabo el proyecto.

Rol	Responsabilidades
Jefe de Proyecto	<ul style="list-style-type: none"> ▪ Definición del alcance del proyecto junto con la administración ▪ Identificación y secuenciación de tareas ▪ Identificación y asignación de recursos (temporales, materiales y recursos humanos) ▪ Determinación de costes ▪ Supervisión de cada fase (Auditorías) ▪ Supervisión del contenido y aspecto de la documentación de cada fase e integración de la documentación final junto con los escritores técnicos
Administración	<ul style="list-style-type: none"> ▪ Definición del alcance del proyecto junto con el jefe de proyecto
Analista	<ul style="list-style-type: none"> ▪ Identificación y análisis de riesgos ▪ Identificación de stakeholders ▪ Análisis del estado del arte ▪ Análisis de requisitos ▪ Análisis de casos de uso ▪ Descripción de la arquitectura (identificación de la tecnología e interfaces)

Sigue en la página siguiente.

Rol	Responsabilidades
Diseñador	<ul style="list-style-type: none"> ■ Descripción de la arquitectura (Diagramas) ■ Diseño de la arquitectura ■ Casos de uso de diseño ■ Diagramas de secuencia de diseño ■ Modelo de dominio de diseño
Programador	<ul style="list-style-type: none"> ■ Implementación de la aplicación
Personal de pruebas	<ul style="list-style-type: none"> ■ Realización de pruebas sobre los módulos y la aplicación completa
Escritores técnicos	<ul style="list-style-type: none"> ■ Elaboración de la documentación de cada fase

Tabla 3.1.: Roles y responsabilidades

Todos los roles serán ocupados por el alumno exclusivamente, al tratarse este proyecto de un Trabajo Fin de Grado, excepto la responsabilidad de la definición del alcance del proyecto asociada al jefe de proyecto, que será compartida entre el alumno y el tutor.

3.3.4. Métodos de trabajo

El conjunto de fases en las que este proyecto estará dividido tomará como referencia el estándar UPEDU (Unified Process for Education). Las fases se subdividen, a su vez, en una serie de actividades que generan distintos artefactos. En un proyecto de software real, cada una de

estas actividades sería asignada a una serie de roles diferentes, contando con diferentes costes que repercuten directamente en el presupuesto del proyecto.

Cada fase termina en un hito, en el cual se habrán generado ciertos artefactos clave para la finalización del proyecto, de entre los descritos en el apartado 3.2.3. Estos entregables serán revisados por el propio alumno y por el tutor responsable del proyecto. El fin de estas revisiones es el de garantizar la calidad e integridad del proyecto, junto con el de encontrar errores en fases tempranas, de manera que las medidas correctivas se ejecuten cuanto antes y las penalizaciones en términos de tiempo y coste sean mínimas.

Las fases que se definen en este análisis podrán, sin embargo, sufrir modificaciones para adaptarse, tanto a las diferentes circunstancias que rodeen a los actores implicados, como a las particularidades de las tecnologías utilizadas y que sean aún desconocidas. No será necesario un protocolo formal para realizar estos cambios, debido al pequeño tamaño del equipo, pero sí deberán quedar convenientemente registrados en este documento.

La comunicación entre los dos actores implicados en el proyecto se realizará, principalmente, mediante el uso de correo electrónico. En los momentos en los que sea necesaria una reunión presencial se utilizará también este medio para concretar la agenda del encuentro.

3.4. Proceso de gestión

3.4.1. Puesta en marcha del plan

La puesta en marcha de un proyecto como este se deberá realizar necesariamente mediante una reunión conjunta entre el tutor y el alumno. En dicha reunión se deberá hacer un primer esbozo del alcance del proyecto y los objetivos, que nos servirán para realizar la identificación de tareas y recursos necesarios.

Podemos, posteriormente, realizar la secuenciación y la planificación temporal de las actividades, que deberán ser elaboradas necesariamente con la experiencia personal como criterio principal, al carecer de información formalizada sobre proyectos similares realizados por el mismo alumno en el pasado. Las tareas serán realizadas íntegramente por el alumno, por lo que no se requerirá hacer una asignación de estas, y el proyecto podrá comenzar en el momento en el que estén definidas.

La identificación y adquisición de las herramientas software o hardware necesarias será realizada por el alumno, bajo la aprobación, ayuda y supervisión del tutor. Estas herramientas se detallan en el apartado 3.5.1. Dependiendo de las herramientas seleccionadas se deberá estimar también el tiempo de formación necesario para dominarlas que necesitará el alumno.

3.4.2. Plan de trabajo

Actividades

Todo lo referente a las actividades necesarias para llevar a cabo el proyecto, así como su descripción, su planificación temporal, los recursos que cada una tiene asignados o la secuenciación entre ellas viene reflejado tanto en las siguientes tablas como en el fichero Project.

En la tabla 3.2 se muestran las actividades relativas a la fase de elaboración del plan del proyecto, en cuyo capítulo nos encontramos.

Las actividades de la siguiente fase comprenden el análisis del proyecto y la investigación del estado del arte. Se detallan en la tabla 3.3.

La siguiente fase comprende el diseño del proyecto, incluyendo la elección de las herramientas software y hardware adecuadas. Sus actividades están contenidas en la tabla 3.4.

La siguiente fase comprenderá tanto la implementación y la realización de pruebas presentadas en las tablas 3.5, 3.6 y 3.7, como la documentación y revisiones finales, contenidas en la 3.8.

Tabla 3.2.: Actividades de la fase de planificación

Id	Nombre de tarea	Descripción	Predecesoras	Duración
1	Plan de desarrollo del proyecto			48 horas
2	Redacción de la documentación	Elaboración de la documentación referente al plan de desarrollo del proyecto	3CC	40 horas
3	Determinar el alcance	Determinación del alcance y los límites		2 horas
4	Planificación			21 horas
5	Definición de actividades	Identificación y descripción de las actividades necesarias para llevar a cabo el proyecto	3	6 horas
6	Secuenciación de actividades	Establecer la relación entre las diferentes actividades	5	4 horas
7	Definir recursos preliminares	Definir los recursos de los que disponemos (temporales, materiales, recursos humanos)	6	4 horas
8	Estimación de duración	Determinación del tiempo en el que se puede llevar a cabo cada tarea	7	3 horas
9	Afianzar Recursos	Asignación de los recursos a cada actividad	8	4 horas
10	Estimación de costes	Cálculo de los costes que supondrá la elaboración del proyecto	4FC+5 horas	5 horas
11	Plan de control del proyecto	Elaboración del plan que se va a seguir para controlar que el proyecto se desarrolla correctamente	10	5 horas
12	Plan de gestión de configuraciones	Creación del plan correspondiente para la gestión de configuraciones	11	5 horas
13	Identificación y análisis de riesgos	Describir los riesgos a los que nos podemos enfrentar, sus posibles consecuencias y cómo enfrentarles	3	9 horas
14	Auditoría	Supervisión y aprobación del plan de desarrollo del proyecto	12;2;13	5 horas

Tabla 3.3.: Actividades de la fase de análisis

Id	Nombre de tarea	Descripción	Predecesoras	Duración
16	Análisis			55 horas
17	Redacción de la documentación de análisis	Elaboración de la documentación referente al análisis de la aplicación	18CC	50 horas
18	Identificación de stakeholders	Identificar, describir e identificar las necesidades de los stakeholders del sistema	13	3 horas
19	Búsqueda y análisis de documentación	Recopilación de información para afrontar el proyecto teniendo en cuenta las necesidades de los stakeholders	18	2 horas
20	Investigación del estado del arte	Recopilación de información sobre los diferentes métodos para la verificación de locutor y las herramientas disponibles para realizarlo	13	40 horas
21	Análisis de requisitos			11 horas
22	Requisitos de usuario	Identificación y descripción de los requisitos de usuario	13FC+5 horas	2 horas
23	Requisitos funcionales	Identificación y descripción de los requisitos funcionales	22	5 horas
24	Requisitos no funcionales	Identificación y descripción de los requisitos no funcionales	22FC+5 horas	3 horas
25	Requisitos de información	Identificación y descripción de los requisitos de información	22FC+8 horas	1 hora
26	Análisis de casos de uso			34 horas
27	Identificación y realización de casos de uso	Desarrollo de los casos de uso del sistema	21	15 horas
28	Matriz de trazabilidad	Elaboración de la matriz de trazabilidad que relacion requisitos y casos de uso	27	2 horas
29	Modelo de dominio	Realización y documentación del modelo de dominio	27FC+6 horas	5 horas
30	Diagramas de secuencia	Realización y documentación de los diagramas de secuencia	29	8 horas
31	Auditoría	Supervisión y aprobación del análisis del proyecto	17;19;26	5 horas

Tabla 3.4.: Actividades de la fase de diseño

Id	Nombre de tarea	Descripción	Predecesoras	Duración
33	Diseño			71 horas
34	Realización de documentación	Elaboración de la documentación referente al diseño de la aplicación	36CC	60 horas
35	Descripción de la arquitectura			14 horas
36	Identificación de la tecnología	Identificación y búsqueda de información acerca de la tecnología necesaria para el desarrollo del proyecto	31	5 horas
37	Identificación de interfaces	Identificación y documentación de las interfaces necesarias	36	3 horas
38	Realización de diagramas	Realización de los diagramas correspondientes a la descripción de la arquitectura (diagrama de despliegue, diagrama de componentes y diagrama de capas)	37	6 horas
39	Diseño de la arquitectura			12 horas
40	Descomposición en subsistemas	Identificación y descripción de los subsistemas que forman el sistema junto con la elaboración del diagrama correspondiente	35	4 horas
41	Topología del sistema	Realización de la ilustración que muestre la topología del sistema	40	1 hora
42	Gestión de la persistencia	Descripción de la tecnología que se utilizará para manejar la persistencia de la aplicación	38FC+5 horas	4 horas
43	Vista lógica del sistema	Elaboración del diagrama que muestra la vista lógica del sistema	36;41FC+4 horas	3 horas
44	Casos de uso de diseño	Realización de los casos de uso de diseño a partir de los casos de uso obtenidos en el análisis	27;39	15 horas
45	Diagramas de secuencia de diseño	Elaboración de los diagramas de secuencia de diseño en caso de que sea necesario	44	15 horas
³⁴ 46	Modelo de dominio de diseño	Elaboración del modelo de dominio de diseño en caso de que sea necesario	45	10 horas

Tabla 3.5.: Actividades de la fase de implementación

Id	Nombre de tarea	Descripción	Predecesoras	Duración
49	Implementación			95 horas
50	Desarrollo del código	Elaboración de los módulos	46FC+6 horas	80 horas
51	Pruebas de los programadores (60%)	Pruebas realizadas sobre los módulos en desarrollo	50FC-80%	40 horas
52	Integración de módulos	Integración de todos los módulos entregados por los diferentes programadores	51;50	10 horas
53	Auditoría	Supervisión y aprobación de los módulos implementados	52	5 horas

Tabla 3.6.: Actividades de la fase de pruebas (módulos)

Id	Nombre de tarea	Descripción	Predecesoras	Duración
55	Pruebas			80 horas
56	Desarrollo del plan de pruebas de caja blanca	Elaboración de la documentación donde se detalla el plan que se seguirá a la hora de realizar las pruebas de caja blanca	46	3 horas
57	Desarrollo del plan de pruebas de caja negra	Elaboración de la documentación donde se detalla el plan que se seguirá a la hora de realizar las pruebas de caja negra	46	3 horas
58	Pruebas de caja blanca en módulos			20 horas
59	Realización de pruebas de caja blanca sobre los módulos		56;54	10 horas
60	Identificar anomalías	Identificación y análisis de los posibles resultados erróneos obtenidos en las pruebas	59	3 horas
61	Revisar y modificar el código		60	7 horas
62	Pruebas de caja negra en módulos			20 horas
63	Realización de pruebas de caja negra sobre los módulos		54;57	10 horas
64	Identificar anomalías	Identificación y análisis de los posibles resultados erróneos obtenidos en las pruebas	63	3 horas
65	Revisar y modificar el código		64	7 horas

Tabla 3.7.: Actividades de la fase de pruebas (programa completo)

Id	Nombre de tarea	Descripción	Predecesoras	Duración
66	Pruebas de caja blanca en el programa completo			20 horas
67	Realización de pruebas de caja blanca sobre el programa completo		58	10 horas
68	Identificar anomalías	Identificación y análisis de los posibles resultados erróneos obtenidos en las pruebas	67	3 horas
69	Revisar y modificar el código		68	7 horas
70	Pruebas de caja negra en el programa completo			20 horas
71	Realización de pruebas de caja negra sobre el programa completo		62	10 horas
72	Identificar anomalías	Identificación y análisis de los posibles resultados erróneos obtenidos en las pruebas	71	3 horas
73	Revisar y modificar el código		72	7 horas
74	Auditoría		73	5 horas

Tabla 3.8.: Actividades finales de documentación

Id	Nombre de tarea	Descripción	Predecesoras	Duración
76	Documentación			55 horas
77	Documentar desarrollo del proyecto	Elaboración de la documentación sobre el plan de desarrollo seguido a lo largo del proyecto y sus conclusiones	55	15 horas
78	Integración de la documentación	Integración de toda la documentación obtenida en las diferentes fases del proyecto y de su desarrollo	77	5 horas
79	Revisión de la documentación	Revisión y aprobación de la documentación de todo el proyecto	78FC+5 horas	5 horas
80	Manual de instalación	Documentación de los pasos necesarios para instalar la aplicación	55;77FC+15 horas	10 horas
81	Manual de usuario	Elaboración del manual de usuario	55;77FC+25 horas	10 horas
82	Presentación powerpoint	Elaboración de la presentación para la defensa	81	5 horas
83	Auditoria	Supervisión y aprobación de toda la documentación del proyecto junto con los manuales	79;81;80	5 horas
84	Entrega documentación		83	0 horas
85	Auditoría de todo el proyecto	Supervisión y aprobación del proyecto	76	10 horas
86	Gestión de cambio	Durante todo el desarrollo del proyecto, éste estará abierto al cambio, los cuales hay que registrar. Esta tarea es la encargada de representar estos posibles cambios	2CC;85FF	5 horas
87	Entrega final	Entrega de la documentación	85	0 horas
88	Defensa ante el tribunal	Defensa del proyecto ante el tribunal designado al efecto	87	0 horas

En el fichero de MS Project incluido en el CD se puede observar con una estructura más detallada la secuenciación de actividades. Todas ellas, excepto las relativas a investigación y definición de los límites del proyecto, se asignarán a una única persona: el alumno.

Este hecho tiene una gran repercusión en la secuenciación de actividades. El proceso unificado está diseñado para que las distintas fases se solapen, haciendo que no sea necesario haber acabado todas las tareas de una fase para comenzar alguna tarea de la siguiente. Este comportamiento es complejo de reproducir con un sólo integrante en el grupo de trabajo del proyecto, por lo que se ha optado por, en los casos en los que es posible, intercalar las actividades de distintas fases en lugar de solaparlas.

Se producen, en ocasiones, sobrecargas en el recurso asignado a todas las tareas, el alumno, y esto es debido a que se ha considerado que tareas distintas, como la implementación y las pruebas, se deben realizar simultáneamente. Esa sobrecarga simula este comportamiento.

Asignación de costes

Uno de los objetivos de este proyecto es el de utilizar, en todos los ámbitos de su elaboración, software gratuito, o que el alumno pueda obtener gratuitamente. Un subconjunto del software utilizado, como Microsoft Project 2013 o Astah Professional, es de pago, pero se ha obtenido gratuitamente a través de las licencias concedidas a la Universidad de Valladolid y sus alumnos. De esta manera, el gasto en software es inexistente.

A nivel de infraestructura, se han utilizado un PC con Windows y un equipo portátil con Mac OS, sin embargo, estos equipos no se han adquirido para la realización de este proyecto, sino que se adquirieron anteriormente y se han reutilizado para el propósito de la realización de este trabajo. No se consideran, por tanto, gastos de adquisición ni de amortización. A la hora de alojar la estructura del proyecto, de forma que sea accesible mediante una conexión a internet, se utilizará, o bien un equipo proporcionado por el alumno, o bien alguna alternativa gratuita, entre las que figuran las que proporciona la E.T.S. de Ingeniería Informática de la Universidad de Valladolid.

Por último, tenemos que analizar los costes a nivel de personal que se derivarían de la realización de este sistema, en el caso de que fuera desarrollado por una entidad privada, y no como un ejercicio académico. Debemos analizar la cantidad de horas que llevará el desarrollo completo, y el sueldo medio (estimado para España) de una persona con la formación necesaria para realizar cada tarea, se describe en la tabla 3.9.

La suma de los costes de personal asciende a 5660€, dado que el resto de costes se han decidido inexistentes, esta es la cifra total asociada al trabajo que se va a desarrollar.

Tabla 3.9.: Costes de personal

Fase	Duración total (h)	Personal necesario	Coste/Hora (€)	Total (€)
Plan de desarrollo	48	Jefe de proyecto	25	960
Análisis	55	Analista software	18	990
Diseño	71	Arquitecto software	20	1420
Implementación	95	Programador	14	1330
Pruebas	80	Programador	12	960

3.4.3. Plan de control del proyecto

Gestión de requisitos

La gestión de requisitos se realizará de manera sencilla. Cualquier cambio en los requisitos, si es de gran importancia para el proyecto, será comunicado al tutor del proyecto para contar con su opinión y permiso. Si no es así, el alumno dispone de libertad para realizarlo.

Las solicitudes de cambio, al tratarse de un equipo de solamente dos personas, no estarán formalizadas, sino que la comunicación por correo electrónico entre las partes implicadas será totalmente suficiente.

Control de la planificación temporal

La planificación temporal será controlada por los hitos que marcan el final de cada fase. No obstante, se tendrán en cuenta las holguras que encontramos hasta el momento de la entrega final del proyecto en Julio, por lo que el control no será altamente restrictivo.

Control de recursos

La asignación de recursos es sencilla. No así concretar la disponibilidad de estos, ya que las cambiantes circunstancias que pueden afectar a los miembros del proyecto durante la duración de este, hacen que este apartado se deba tratar con cautela.

Informes y plan de comunicaciones

La principal vía de comunicación entre los dos miembros del equipo será el correo electrónico. Se producirán también reuniones en los momentos en los que se haya acumulado una carga de trabajo que aconseje poner en común el trabajo realizado. Normalmente, estas reuniones coincidirán con los hitos existentes en la planificación.

No se va a utilizar ningún tipo de informe para la solicitud de reuniones o cambios en la documentación, debido a la facilidad de lograr una comunicación pequeña en el pequeño grupo de trabajo con que este proyecto cuenta.

Plan de medición

Hay dos medidas que se podrán tener en cuenta durante la realización del proyecto.

- **Duración:** La duración de las tareas, que puede o no coincidir con la establecida en la planificación (dependiendo de los riesgos, de una mala estimación, ...). Esta duración se presenta en horas, que se repartirán durante los días de margen para la realización del proyecto, siendo unas 6 horas por semana de media, la cantidad de trabajo habitual.
- **Calidad de los entregables:** Incluyendo la calidad tanto de documentos como de código.

Medidas como el esfuerzo de los miembros del equipo o los costes no proceden para el proyecto actual.

La evaluación de estas medidas correrá a cargo, en primera instancia, del tutor del proyecto. Todas las medidas deberán superar unos requisitos de calidad para entregar el proyecto en la fecha planteada por la dirección de la escuela.

El estado de estas mediciones se tratará mediante la comunicación entre alumno y tutor, ya sea por correo electrónico o de forma presencial.

3.4.4. Plan de riesgos

Se listan los riesgos detectados en el proyecto, distribuidos en las tablas 3.10, 3.11, 3.12, 3.13 y 3.14.

La medida de la exposición se ha calculado utilizando la norma ISO/IEC 27005:2008, reflejada en la figura 3.1.

Figura 3.1.: ISO/IEC 27005:2008: Estándar utilizado para identificar la exposición al riesgo

		Likelihood of Incident Scenario				
		Very Low	Low	Medium	High	Very High
Business Impact	Very Low	0	1	2	3	4
	Low	1	2	3	4	5
	Medium	2	3	4	5	6
	High	3	4	5	6	7
	Very High	4	5	6	7	8

Tabla 3.10.: Riesgo 01 - Cambio del análisis

Número: 01	Nombre del riesgo: Cambio del análisis de la aplicación	Fecha: 20-03-2014
Descripción del riesgo		
Cambio de requisitos en el proyecto, habiendo realizado parte del análisis para los que se tenían.		
Causas		
<ul style="list-style-type: none"> ■ Fallo de comunicación entre el tutor y el alumno <ul style="list-style-type: none"> ■ Cambio de tecnología ■ Errores en la fase de investigación del estado del arte 		
Consecuencias		
Se producirá un retraso en el proyecto para corregir las secciones del análisis afectadas.		
Plan de acción		
Estrategia: Protección del riesgo	Descripción: Se hará una revisión continua del trabajo realizado hasta el momento. Se probarán las tecnologías lo antes posible durante la fase de desarrollo.	
Impacto: Alto	Probabilidad: Baja	Exposición: 4

Tabla 3.11.: Riesgo 02 - Reducción de la disponibilidad de un miembro del equipo

Número: 02	Nombre del riesgo: Reducción de la disponibilidad de algún integrante del proyecto	Fecha: 20-03-2014
Descripción del riesgo		
Cambian las circunstancias del alumno o tutor y estos no pueden dedicar el tiempo suficiente al proyecto.		
Causas		
<ul style="list-style-type: none"> ▪ Enfermedad ▪ Accidente ▪ Obligaciones contractuales 		
Consecuencias		
La/s tarea/s que esa persona tuviera que realizar, deberá/n ser alargada/s en el tiempo hasta poder realizarse correctamente		
Plan de acción		
Estrategia: Reservar el riesgo	Descripción: En caso de que sea el alumno el afectado por este riesgo, no quedará otro remedio que retrasar el desarrollo del proyecto en la medida en que sea necesario, pudiendo llegar, incluso, a retrasar la fecha de entrega. La probabilidad de este riesgo, en este caso, es alta, ya que la realización de las prácticas en empresa de la carrera deberá coincidir con la realización del proyecto, de tal manera que, si se prolongan en el tiempo a causa de un posterior contrato o resultan ser de un alto número de horas, la planificación del proyecto se va a ver seriamente afectada. Si es el tutor el afectado, algo menos probable, el alumno seguirá con el trabajo normal, a la espera de que, las dudas que le surjan, puedan ser respondidas por el tutor cuando se encuentre disponible.	
Impacto: Muy Alto	Probabilidad: Alto	Exposición: 7

Tabla 3.12.: Riesgo 03 - Fallo en los equipos

Número: 03	Nombre del riesgo: Fallo de los equipos de desarrollo	Fecha: 20-03-2014
Descripción del riesgo		
Aparición de fallos de cualquier índole en uno o varios de los equipos que usan los integrantes del proyecto		
Causas		
<ul style="list-style-type: none"> ■ Fallos en el hardware de los equipos ■ Infección por malware 		
Consecuencias		
Implicará un retraso del proyecto, al tener que subsanar los problemas surgidos en las máquinas		
Plan de acción:		
Estrategia: Protección del riesgo / Aceptación del riesgo	Descripción: El plan de acción de este riesgo consiste en disminuir la probabilidad de aparición del mismo a causa de malware, teniendo un antivirus actualizado. En el caso de fallos hardware, se deberá cuidar que la temperatura del equipo sea adecuada, limpiándolo en caso contrario.	
Impacto: Bajo	Probabilidad: Baja	Exposición: 2 (*)

Tabla 3.13.: Riesgo 04 - Poca destreza con las herramientas

Número: 04	Nombre del riesgo: Poca destreza con las herramientas	Fecha: 20-03-2014
Descripción del riesgo		
Aumento del tiempo necesario para la finalización del proyecto		
Causas		
<ul style="list-style-type: none"> ▪ Poca familiarización de los miembros del equipo con las herramientas de desarrollo 		
Consecuencias		
Se producen retrasos en la entrega del proyecto		
Plan de acción		
Estrategia: Evitación del riesgo	Descripción: Se planificarán tareas de formación e investigación en los momentos en que sea necesario y, para que estas no se extiendan demasiado, se procurará utilizar tecnologías conocidas.	
Impacto: Medio	Probabilidad: Baja	Exposición: 3

Tabla 3.14.: Riesgo 05 - Estimaciones imprecisas

Número: 05	Nombre del riesgo: Estimaciones imprecisas	Fecha: 20-03-2014
Descripción del riesgo		
A causa de una mala planificación del proyecto, las estimaciones de uso de recursos y división en tareas, no son las adecuadas para el desarrollo correcto del proyecto		
Causas		
<ul style="list-style-type: none"> ▪ Demasiados hitos/tareas en la planificación ▪ Pocos hitos/tareas en la planificación ▪ Deficiente gestión de los recursos 		
Consecuencias		
La consecuencia es el fracaso del proyecto, si no se reacciona a las malas estimaciones a tiempo		
Plan de acción		
Estrategia: Protección del riesgo	Descripción: Se deberá llevar un seguimiento de la planificación para poder tener una reacción temprana a las incongruencias que se vayan pudiendo detectar, redistribuyendo los recursos o añadiendo/eliminando ciertas tareas/hitos	
Impacto: Alto	Probabilidad: Medio	Exposición: 5 (*)

Tabla 3.15.: Riesgo 06 - Problemas con el uso de las herramientas

Número: 06	Nombre del riesgo: Problemas con el uso de las herramientas	Fecha: 27-06-2014
Descripción del riesgo		
Se van a utilizar herramientas experimentales a la hora de realizar la verificación biométrica. Esto implica que el software utilizado puede contener bugs, o su funcionamiento puede no alcanzar unas cuotas de calidad mínimas.		
Causas		
<ul style="list-style-type: none"> ▪ Bugs en el software de terceros ▪ Funcionamiento deficiente para el dominio del proyecto 		
Consecuencias		
Se producirá un retraso en la planificación del proyecto, cuya duración vendrá determinada por el tiempo que se necesite para corregir las deficiencias presentadas o cambiar el software utilizado.		
Plan de acción		
Estrategia: Protección del riesgo	Descripción: A la hora de escoger el sistema de reconocimiento de locutor a utilizar, se deberán plantear varias alternativas, escogiendo la solución más madura, documentada y compatible con nuestra plataforma. También se deben conocer las diferentes opciones disponibles, dentro del software elegido, para realizar el proceso de verificación de locutor, lo que permitirá una readaptación lo más rápida posible de las especificaciones del proyecto.	
Impacto: Alto	Probabilidad: Medio	Exposición: 5 (*)

3.4.5. Plan de cierre

Para concluir este proyecto, se realizará la integración de toda la documentación recogida a lo largo del mismo, en esta integración se realizará también una revisión, junto con todo ello se incluirá una evaluación de las medidas que se han ido tomando y las conclusiones correspondientes. Se elaborarán también los manuales tanto de instalación como el de ayuda al usuario. Con lo que la documentación quedará totalmente finalizada.

Posteriormente, será el tutor el encargado de de realizar una auditoría general de toda la documentación elaborada y del código desarrollado.

Tras la elaboración del proyecto se realizará una presentación ante el tribunal encargado de evaluarlo. Esta defensa aparece en la planificación temporal presentada en Project y en el apartado 3.4.2.

3.5. Proceso técnico

3.5.1. Métodos, herramientas y técnicas

Se toma como referencia el modelo de proceso unificado que desarrolla UPEDU[Mon14]. No obstante, su aplicación no será estricta, debido a las limitaciones que, tanto esta como la gran mayoría de metodologías de desarrollo, presentan para adaptarse a un proyecto con un solo integrante.

Sí que se aplicará, sin embargo, la división en fases propuesta por UPEDU: Iniciación, Elaboración, Construcción y Transición. Estas fases se definen de forma detallada en la tabla 3.16

Una vez identificado el modelo de trabajo a seguir, es necesario identificar con qué más elementos se va a trabajar para elaborar las diferentes entregas del proyecto.

Documentación

En primer lugar nos encontramos con la realización de la documentación, la cual se elaborará a lo largo de las diferentes fases, necesitando realizar un proceso de integración de toda ella en la etapa final.

Toda la documentación será elaborada en \LaTeX , utilizando para su compilación la herramienta PDF \LaTeX . Para cumplir los criterios de calidad, se utilizarán diversas plantillas:

- **Software Project Plan Template - IEE 1058 - 1998 - ISO 12207:** Plantilla utilizada para la elaboración de este capítulo.
- **OpenUP:** Las fases de análisis y diseño utilizarán las plantillas que proporciona OpenUP, con ciertas modificaciones que las hagan más apropiadas a este proyecto.

Análisis y Diseño

Será necesaria la creación de diversos diagramas para completar las fases de análisis y diseño de la aplicación, para lo cual se utilizará Astah Professional.

Tabla 3.16.: Fases en UPEDU

Fase	Descripción
Fase de Iniciación	Fase en la que se identifican los objetivos de la elaboración del proyecto presentados en el apartado 3.2.1, así como el plan de desarrollo de este. La finalización de esta fase produce el artefacto “Plan de desarrollo”.
Fase de Elaboración	Identificación de requisitos (funcionales, no funcionales y de información) y análisis de los mismos. Como resultado de esta fase se obtendrán también los casos de uso y diagramas de secuencia tanto desde el punto de vista del análisis como desde el punto del diseño. El final de esta fase produce los artefactos “Documento de análisis” y “Documento de diseño”. Upedu indica la realización de dos iteraciones para esta fase. En nuestro caso, debido al tiempo disponible, no se puede asegurar la realización de estas iteraciones, por lo que el tiempo no utilizado en la segunda iteración será empleado, bien para la realización de la siguiente fase o para proporcionar holgura a las tareas de la fase en la que nos encontramos.
Fase de Construcción	Se realiza la implementación de la aplicación. En este caso, a pesar de no quedar representado en el fichero Project, se pretende realizar al menos dos iteraciones (UPEDU indica la realización de tres). En caso de no disponer de tiempo, al igual que en el caso anterior, las tareas incluidas en esta fase dispondrán de una mayor holgura. Upedu, además, incluye en esta etapa la realización de las pruebas sobre la aplicación.
Fase de Transición	Preparación del software para la utilización del usuario final. Incluye la creación de manuales de instalación y de usuario. Dará lugar a los artefactos finales que se entregarán para la corrección del tribunal del Trabajo de fin de Grado. Debería incluirse, entre otras, la realización de fases beta en la que se pruebe la aplicación en el escenario del usuario final, lo cual no es posible dadas las circunstancias.

Implementación

En esta sección se listarán todas las características de la implementación, desde lenguaje de programación hasta material hardware necesario.

- Sistema Operativo: Linux, para la parte web y el servidor. Android para la aplicación móvil.
- Lenguaje de programación: PHP para la parte web. Android (junto con el SDK de Android) para la aplicación móvil.
- Base de datos: MySQL.
- Servidor: Apache.
- IDE: Eclipse.

Otras herramientas

Ha sido necesaria también la utilización de Microsoft Project, en su versión de 2013, para llevar a cabo la planificación que aparece adjunta con este documento, y la de MySQL Workbench, para crear el modelo entidad-relación de la base de datos y los scripts SQL asociados.

Capítulo 4.

Análisis

4.1. Introducción

Se presenta en el siguiente capítulo la elaboración del análisis de la aplicación a desarrollar, siguiendo, aunque no de manera estricta, la plantilla proporcionada por OpenUp. Para la elaboración de diagramas se utilizará la herramienta Astah Professional 6.8.0.

4.1.1. Propósito del sistema

El propósito del sistema será facilitar a los usuarios interesados el uso de un sistema de verificación de locutor de carácter experimental, es decir, a modo de demostrador y como plataforma de prueba de los sistemas desarrollados por el GIR ECA-SIMM. Para ello se desarrollará tanto una página web como una aplicación móvil que utilizarán un software de terceros para todo lo concerniente con el reconocimiento biométrico mediante voz, es decir, se encargará de obtener las características del audio y realizar las comparaciones entre los modelos de distintas muestras.

4.1.2. Planteamiento del problema y objetivos

El uso de técnicas biométricas para lograr la autenticación de usuarios es algo en lo que se lleva investigando desde hace muchos años, con cada vez más aplicaciones prácticas. A pesar de esto, los últimos avances llevados a cabo en las tecnologías web, con la aparición de HTML5 o nuevas librerías para JavaScript, y la proliferación de los dispositivos móviles y su gran cantidad de sensores amplían la facilidad y calidad de acceso este tipo de proyectos.

El objetivo de este proyecto consiste en, tomando ventaja del marco tecnológico en que nos encontramos y utilizando un enfoque orientado a obtener las mejores características de mantenibilidad, extensibilidad, y calidad del código, crear tanto una página web como una aplicación móvil que recojan el audio de una muestra de voz y la envíen a un servidor encargado de procesarla y verificar la identidad del locutor.

Se cuenta con un sistema, desarrollado para un proyecto fin de carrera anterior, que nos servirá, únicamente, para extraer los requisitos necesarios a la hora de realizar el nuevo, ya que el sistema a llevar a cabo en este proyecto se creará desde cero.

Los stakeholders detectados para el proyecto se muestran en la tabla 4.1

Este proyecto debe, en consecuencia, permitir que los stakeholders lleven a cabo sus responsabilidades, aunque no necesariamente se proveerá de una forma automatizada de realizarlas. Sólomente las principales necesidades se transformarán directamente en características del proyecto, de acuerdo a la tabla 4.2.

Tabla 4.1.: Stakeholders del sistema

Nombre	Descripción	Responsabilidades
Administrador del sistema	Encargado de gestionar el sistema	<ul style="list-style-type: none"> ■ Mantener la disponibilidad del sistema ■ Garantizar el funcionamiento del sistema de autenticación biométrica. ■ Atender las peticiones de los usuarios registrados referentes al borrado y modificación de sus datos.
Usuarios finales	Toda persona interesada en el producto final de un proyecto	<ul style="list-style-type: none"> ■ Registrarse en el sistema. ■ Inscribir su voz en el sistema. ■ Verificar la autenticidad de cualquier muestra de audio contra la un usuario inscrito.
Administrador de la base de datos	Encargado de gestionar la base de datos	<ul style="list-style-type: none"> ■ Definir la estructura de la base de datos. ■ Mantener la disponibilidad de la base de datos. ■ Realizar el respaldo de la base de datos. ■ Encargado del mantenimiento y actualización del SGBD. ■ Diccionario de datos.
Investigador	Encargado de optimizar el sistema de verificación de locutor	<ul style="list-style-type: none"> ■ Modificar las configuraciones del sistema biométrico para mejorar los resultados. ■ Mantener el modelo del mundo, regenerarlo cuando sea necesario. ■ Recopilar información estadística sobre el óptimo funcionamiento de la verificación de locutor.

Tabla 4.2.: Necesidades y características

Necesidad	Prioridad	Características
Registro en el sistema	Alta	<ul style="list-style-type: none">■ Identificar al usuario en el sistema■ Permitirle realizar la inscripción de su voz en el sistema
Inscripción de voz en el sistema	Alta	<ul style="list-style-type: none">■ Envío de una muestra de voz al sistema para ser procesada, generando su modelo y almacenándolo.
Verificación de locutor	Alta	<ul style="list-style-type: none">■ Envío de una muestra de voz al sistema para generar su modelo y compararlo con otro, dando un valor de su similitud.

4.2. Especificación de requisitos

4.2.1. Requisitos de usuario

RU01 - El usuario se registrará en el sistema.

RU02 - Los usuarios registrados podrán inscribirse en el sistema (crear su modelo).

RU03 - Un usuario puede verificar una muestra de audio contra la identidad de cualquier usuario inscrito.

Todos estos requisitos son acciones que podrán realizar los actores del sistema, los cuales nos permitirán identificar los requisitos funcionales que se presentan a continuación.

4.2.2. Requisitos funcionales

RF01 - El sistema permitiría registrarse a los usuarios.

RF02 - El sistema identificará a los usuarios por su nombre de usuario único.

RF03 - El sistema grabará el audio del dispositivo en que se utilice.

RF04 - El sistema solicitará al software de reconocimiento de voz que genere el modelo de una muestra de audio.

RF05 - El sistema solicitará al software de reconocimiento de voz que compare dos modelos de dos muestras de audio.

RF06 - El sistema obtendrá los datos que genere el software de reconocimiento de voz como resultado de una grabación y se los mostrará al usuario.

4.2.3. Requisitos no funcionales

RNF01 - La aplicación web correrá sobre un servidor apache 2.2.

RNF02 - La aplicación se construirá utilizando PHP 5.4.4.

RNF03 - Los datos se almacenarán en una base de datos MySQL 5.5.

RNF04 - La aplicación web deberá funcionar en cualquier navegador que soporte WebRTC ¹.

¹En el momento de la redacción de este documento, las últimas versiones estables de los navegadores Chrome, Firefox y Opera, tanto en sus versiones de escritorio como móviles, soportan esta característica

RNF05 - La aplicación Android deberá funcionar en dispositivos con una versión del S.O. igual o superior a la 3.0.

Usabilidad

RNF06 - Un usuario con conocimientos bajos sobre las tecnologías podrá utilizar todas las funcionales de las que dispone el sistema con diez minutos de entrenamiento.

RNF07 - Los mensajes de error presentados en la aplicación deben poder ser inequívocamente identificados por el usuario a través del uso de una descripción adecuada.

Fiabilidad

RNF08 - Se requerirá el uso de transacciones (SGBD) para que la base de datos llegue siempre a un estado coherente.

RNF09 - El servidor web deberá estar disponible 24/7.

Rendimiento

RNF010 - El 95 % de las transacciones en la aplicación que no involucren la transmisión de ficheros de audio deben tardar, como máximo, dos segundos, con una carga del sistema del 80 %.

RNF011 - El 75 % de las transacciones en la aplicación que involucren la transmisión de ficheros de audio de menos de dos minutos deben tardar, como máximo, un minuto y medio, con una carga del sistema del 80 % y una velocidad de subida de internet de, al menos, 500 Kbps.

Mantenibilidad

RNF012 - El sistema deberá estar completamente documentado, tanto en manuales de usuarios, de instalación y de administración como en el código fuente de la aplicación.

RNF013 - El sistema deberá ser lo más simple posible y claro para su futuro mantenimiento.

Seguridad

RNF014 - La inscripción en el sistema sólo puede realizarse para un usuario registrado dado.

RNF015 - La modificación o borrado de datos, tanto de registro como de voz, de un usuario, solo podrá ser realizada por el administrador accediendo directamente a la base de datos.

RNF016 - El sistema contará los siguientes tres ficheros log:

- Biométrico: Guarda todas las acciones realizadas por el sistema de reconocimiento biométrico.
- Accesos: Guarda todos los registros, inscripciones y peticiones de verificación del sistema.
- Errores: Guarda todos los errores de la aplicación.

Verificación de datos

RNF017 - El sistema deberá validar la información que se introduzca en cualquiera de los formularios. Esta validación incluye la obligatoriedad de los campos, el tipo y sus características.

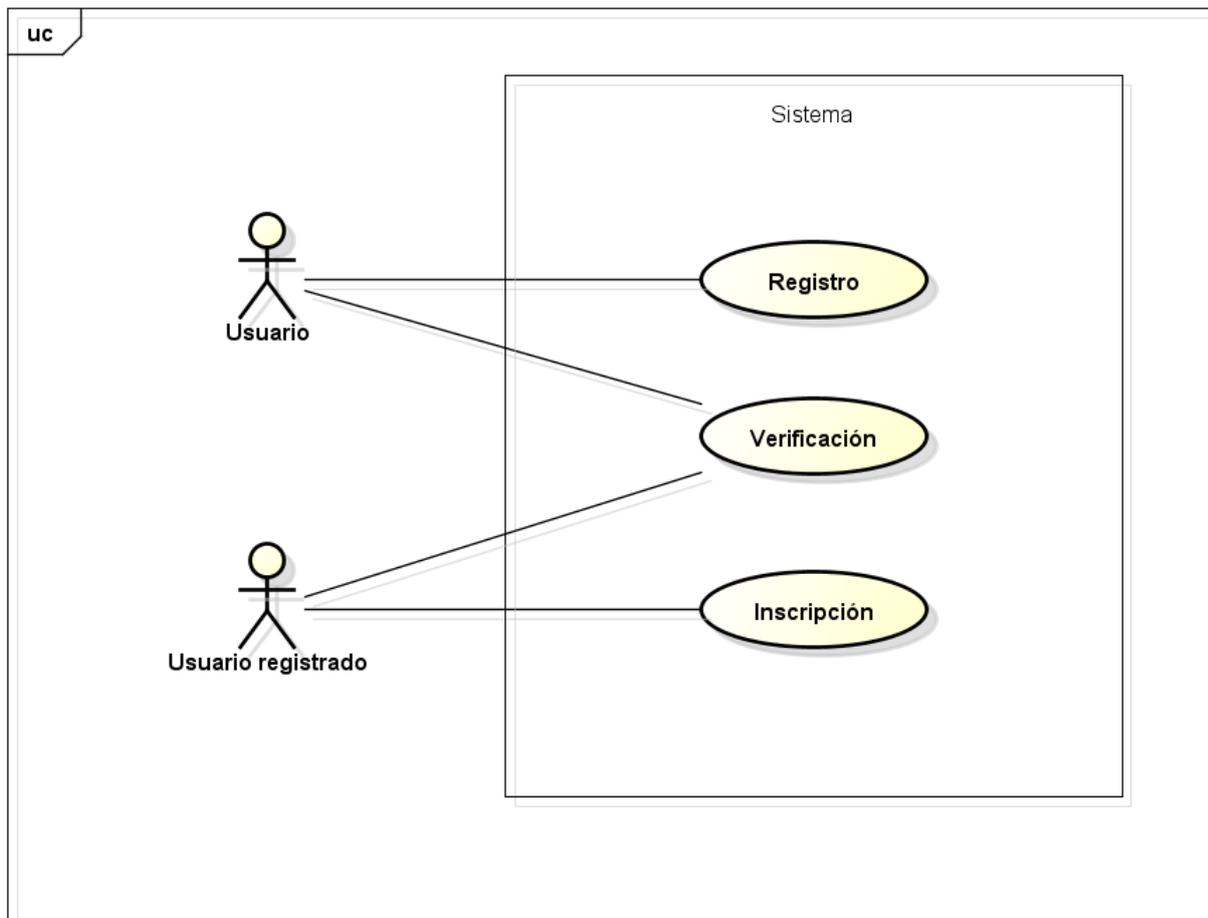
4.2.4. Requisitos de información

RI01 - El sistema guardará información acerca de los usuarios, concretamente, un nombre de usuario, nombre, apellidos, dirección de correo electrónico y fecha y hora del registro.

4.3. Casos de uso

4.3.1. Diagrama de casos de uso

Figura 4.1.: Diagrama de casos de uso



4.3.2. CU01 - Registro en el sistema

Descripción

Un usuario debe registrarse en el sistema para poder realizar la inscripción de su voz.

Actores

- Usuario

Precondiciones

No hay precondiciones

Flujo normal de eventos

1. El caso de uso comienza cuando el usuario solicita registrarse en el sistema.
2. El sistema pide los datos necesarios para crear el nuevo usuario.
3. El usuario otorga las credenciales.
4. El sistema comprueba la validez de los datos introducidos.
5. El sistema crea un nuevo usuario.
6. El caso de uso termina.

Flujo alternativo**Fallo de validez de datos**

Si en el paso 4 el sistema comprueba que al menos uno de los datos no es válido.

1. El sistema muestra esta información al usuario.
2. El caso de uso finaliza con error.

Fallo de creación de usuario

Si en el paso 5 el sistema no consigue crear el usuario.

1. El sistema muestra esta información al usuario.
2. El caso de uso finaliza con error.

Escenarios clave

Roberto, una persona interesada en los sistemas biométricos, accede a este sistema. Como quiere probar qué tal funciona inscribiendo su voz y haciendo diferentes test, decide registrarse. El sistema le pide los datos necesarios para hacer la inscripción, Roberto los proporciona y recibe un mensaje de confirmación indicándole que se ha registrado correctamente. Desde este momento, Roberto puede inscribir su voz en el sistema.

Postcondiciones

Finalización satisfactoria

Se almacenan los datos proporcionados por el usuario.

Finalización con errores

No se almacenarán los datos proporcionados por el usuario.

4.3.3. CU02 - Inscripción en el sistema

Descripción

El usuario desea inscribir su voz en el sistema de autenticación biométrica.

Actores

- Usuario registrado

Precondiciones

- El usuario debe estar registrado en el sistema.

Flujo normal de eventos

1. El caso de uso comienza cuando el usuario solicita realizar la inscripción de su voz.
2. El sistema pide las credenciales al usuario.
3. El usuario introduce sus credenciales en el sistema.
4. El sistema comprueba la validez de las credenciales.
5. El sistema solicita una muestra de audio al usuario.
6. El usuario proporciona una muestra de audio al sistema.
7. El sistema recoge la muestra de audio y la manda procesar al software de autenticación biométrica.
8. El caso de uso termina.

Flujo alternativo**Credenciales no válidas**

Si en el paso 4 el sistema comprueba que las credenciales no son válidas.

1. El sistema informa al usuario mediante un mensaje de error.
2. El caso de uso finaliza.

Usuario ya inscrito

Si en el paso 4 el sistema comprueba que el usuario ya está inscrito.

1. El sistema informa al usuario usuario de este hecho.
2. El caso de uso continúa en el paso 5.

Error al procesar audio

Si en el paso 7 el software de autenticación biométrica no consigue completar el procesamiento del audio.

1. El sistema informa al usuario mediante un mensaje de error.
2. El caso de uso finaliza.

Cancelación

En cualquiera de los pasos el usuario indica que quiere cancelar el caso de uso.

1. El caso de uso finaliza.

Escenarios clave

Un usuario registrado, David, decide probar qué tal funciona la verificación de locutor del sistema. Para ello, indica al sistema que quiere realizar la inscripción proporcionando sus credenciales, el sistema le pide un fichero de audio con su voz y David lo envía. El sistema recoge el fichero de audio y se lo pasa al software de autenticación biométrica para que procese la muestra de voz de David. Se genera su modelo correctamente, por lo que el sistema muestra a David un mensaje de confirmación indicándole el éxito de su inscripción.

Postcondiciones

Finalización satisfactoria

El software de autenticación biométrica generará los datos oportunos para almacenar las características de la voz del usuario inscrito. Se informará por pantalla al usuario del éxito del proceso.

Finalización con errores

Se mostrará un mensaje de error y se almacenarán los datos de la ejecución en el log.

4.3.4. CU03 - Verificación de locutor

Descripción

Un usuario realiza una verificación de locutor.

Actores

- Usuario
- Usuario registrado

Precondiciones

No hay precondiciones

Flujo normal de eventos

1. El caso de uso comienza cuando el usuario solicita realizar una verificación de locutor.
2. El sistema solicita la identificación del usuario contra el que se quiere realizar la verificación.
3. El usuario proporciona la identificación del usuario contra el que se quiere realizar la verificación.
4. El sistema comprueba que la identidad del usuario es válida.
5. El sistema solicita una muestra de audio al usuario.
6. El usuario proporciona una muestra de audio al sistema.
7. El sistema recoge la muestra de audio y pide su verificación contra el usuario elegido al software de autenticación biométrica.
8. El caso de uso termina.

Flujo alternativo

Usuario no registrado

En el paso 4 el usuario introduce la identificación de un usuario no registrado.

1. El sistema indicará mostrará un mensaje informando de que el usuario indicado no existe.
2. El caso de uso termina.

Usuario no inscrito

En el paso 4 el usuario introduce la identificación de un usuario que aún no ha realizado la inscripción.

1. El sistema indicará mostrará un mensaje informando de que el usuario indicado no ha realizado la inscripción.
2. El caso de uso termina.

Error al procesar audio

Si en el paso 7 el software de autenticación biométrica no consigue completar el procesamiento del audio.

1. El sistema informa al usuario mediante un mensaje de error.
2. El caso de uso finaliza.

Cancelación

En cualquiera de los pasos el usuario indica que quiere cancelar el caso de uso.

1. El caso de uso finaliza.

Escenarios clave

Natalia accede al sistema. Hace unos días se registró y realizó la inscripción, así que se dispone a comprobar qué tal funciona la verificación de locutor. Introduce su identificador de usuario en el sistema, que le pide una muestra de audio con su voz. Natalia proporciona dicha muestra de audio, el software de autenticación biométrica la procesa y la compara con los datos que generó y almacenó en el momento en que natalia se inscribió, proporcionando un valor de la similitud entre ambas voces. Este valor se le muestra a Natalia.

A Laura siempre le han dicho que tiene una voz muy parecida a la de Natalia. Natalia le habla de este sistema y Laura decide probar si realmente su voz podría pasar por la de Natalia. Laura nunca se ha registrado en el sistema, pero accede al sistema, introduce el identificador de Natalia y proporciona una muestra de audio de su voz. El sistema la procesa y la compara con los datos que generó para la voz de natalia cuando esta se inscribió, mostrando un valor de la similitud de las dos voces a Laura.

Postcondiciones

Finalización satisfactoria

El software de autenticación biométrica realizará la comparación entre los dos usuarios. Se informará por pantalla al usuario del resultado del proceso.

Finalización con errores

Se mostrará un mensaje de error y se almacenarán los datos de la ejecución en el log.

4.4. Matriz de trazabilidad

Se presenta a continuación la matriz de trazabilidad, la cual relaciona los casos de uso identificados con los requisitos funcionales.

Requisitos funcionales

RF01 - El sistema permitiría registrarse a los usuarios.

RF02 - El sistema identificará a los usuarios por su nombre de usuario único.

RF03 - El sistema grabará el audio del dispositivo en que se utilice.

RF04 - El sistema solicitará al software de reconocimiento de voz que genere el modelo de una muestra de audio.

RF05 - El sistema solicitará al software de reconocimiento de voz que compare dos modelos de dos muestras de audio.

RF06 - El sistema obtendrá los datos que genere el software de reconocimiento de voz como resultado de una grabación y se los mostrará al usuario.

Casos de uso

CU01 - Registro en el sistema.

CU02 - Inscripción en el sistema.

CU03 - Verificación de locutor.

Matriz

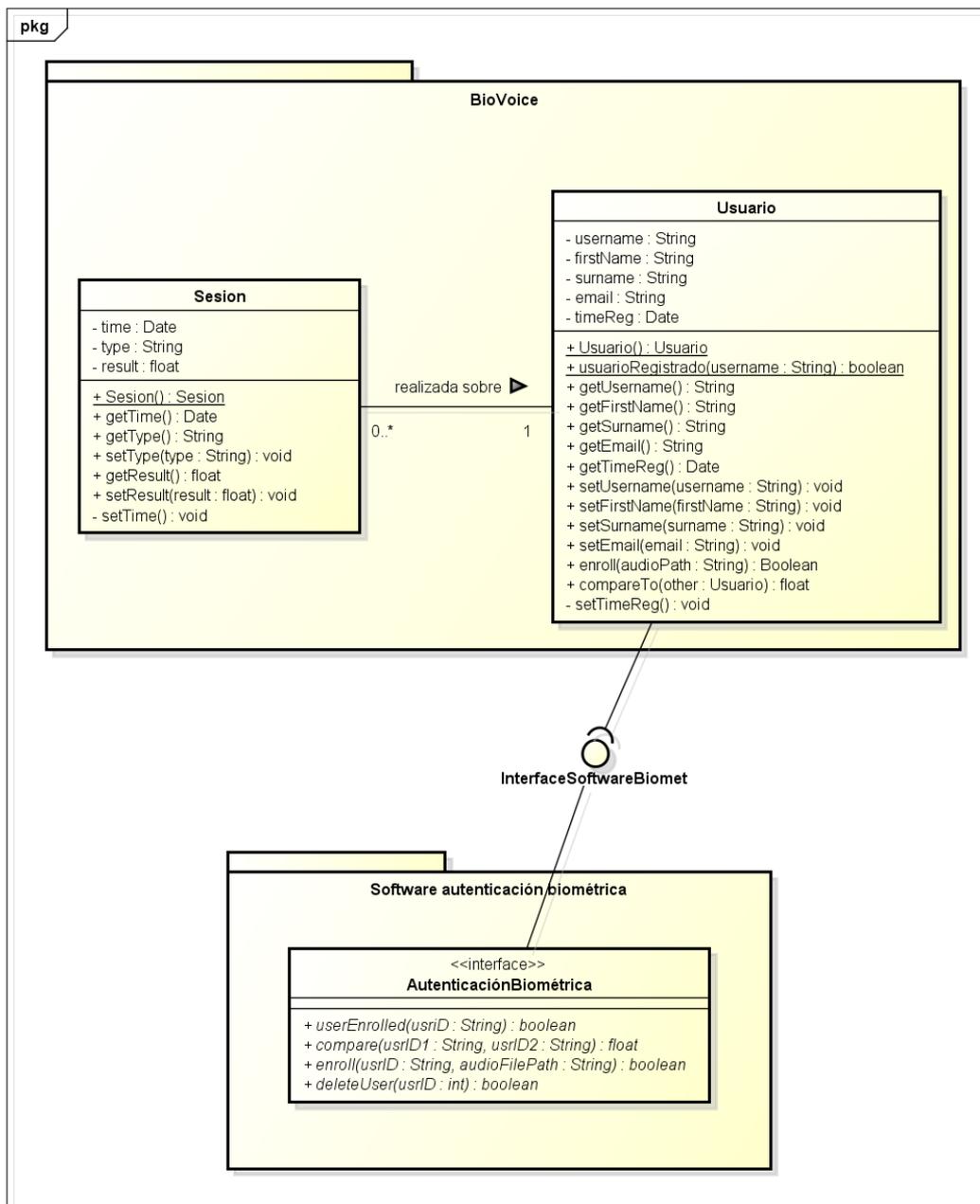
Se busca plasmar en la matriz de trazabilidad si todos los requisitos están cubiertos y dónde.

Tabla 4.3.: Matriz de trazabilidad

	CU01	CU02	CU03
RF01	X		
RF02	X	X	X
RF03		X	X
RF04		X	X
RF05			X
RF06		X	X

4.5. Modelo de dominio

Figura 4.2.: Modelo de dominio

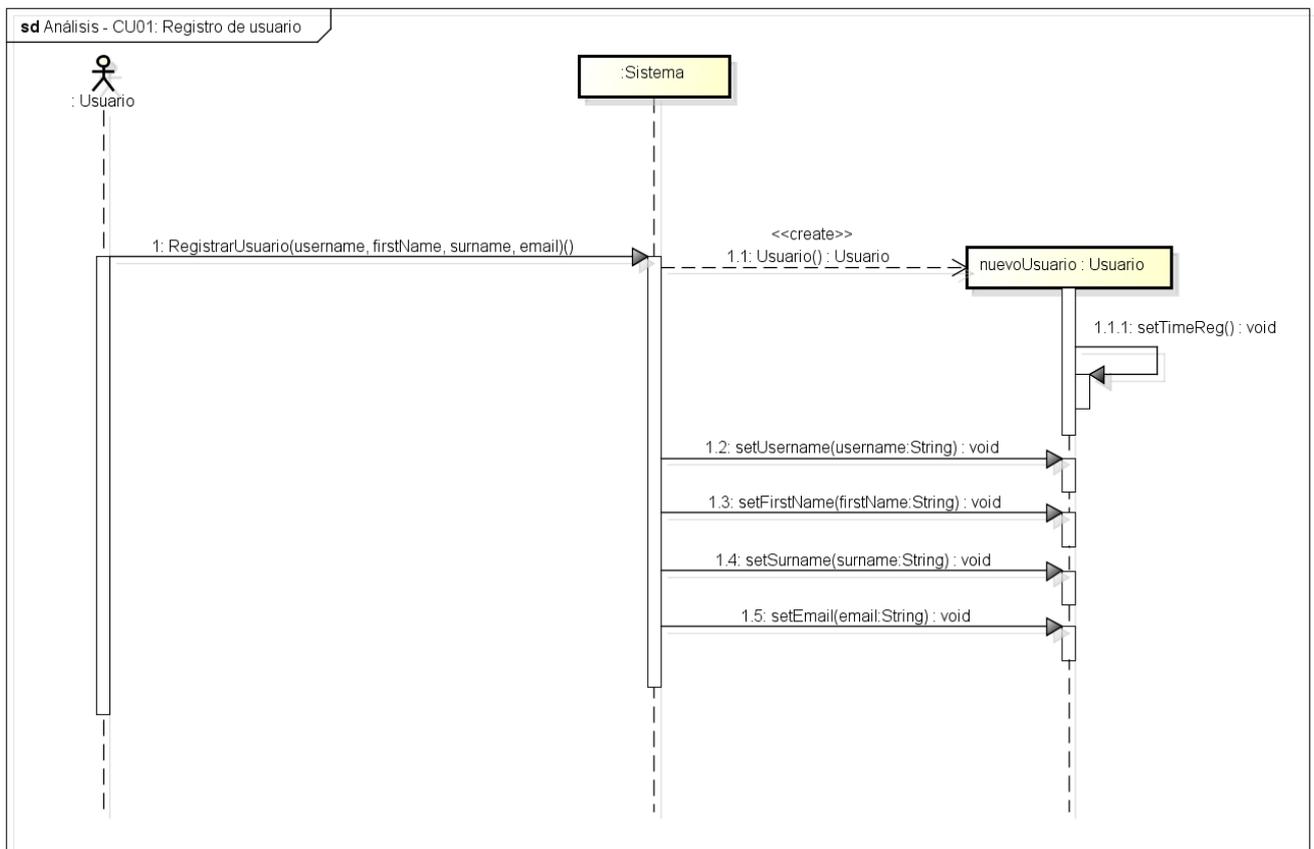


4.6. Diagramas de secuencia

Quedan presentados a continuación los diagramas de secuencia de los distintos casos de uso. Hay que tener en cuenta que el actor usuario representa a los actores correspondientes, indicados anteriormente, en cada caso de uso. Además, dado que la mayoría de casos de uso son un cambio de estado a otro quedarán representados por el diagrama de secuencia “Cambiar estado de una solicitud de cambio”.

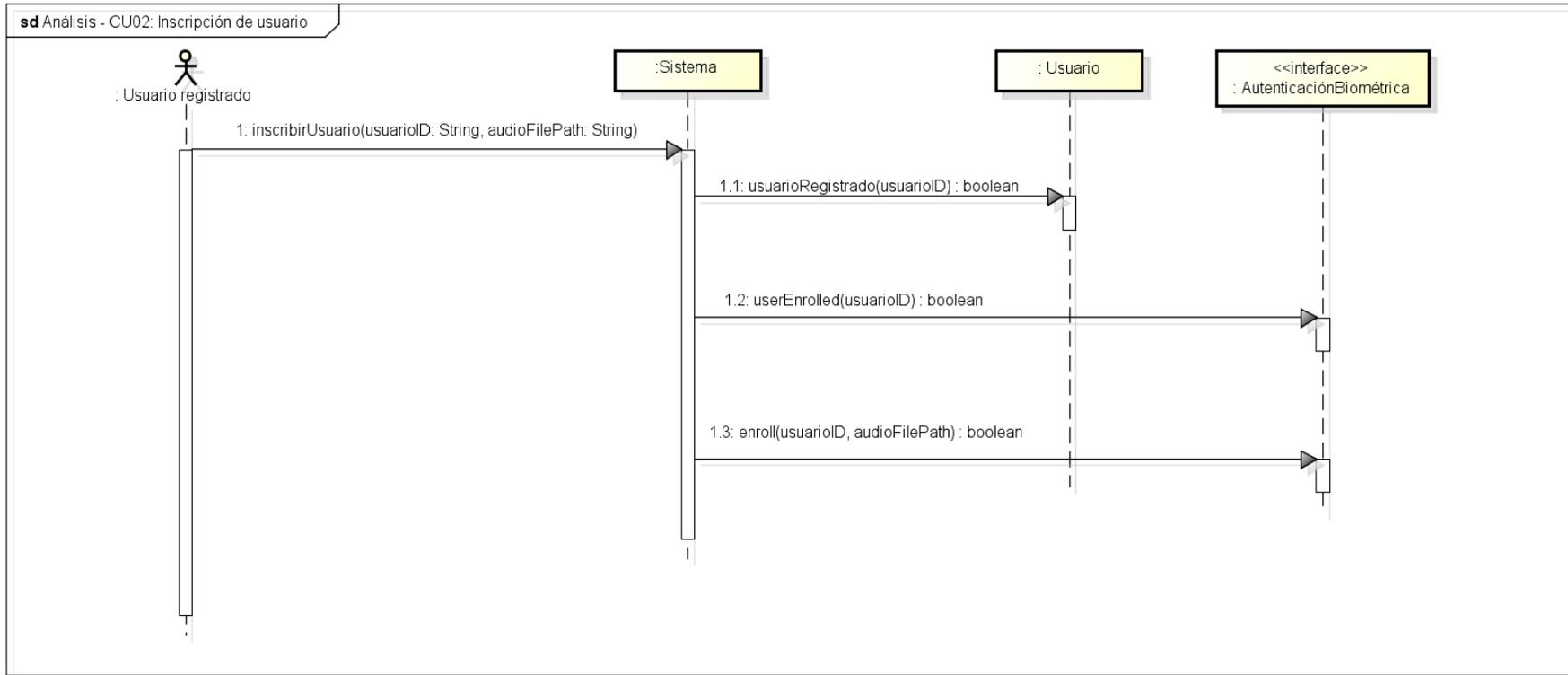
4.6.1. CU01 - Registro de usuario

Figura 4.3.: Diagrama de secuencia CU01 - Registro de usuario



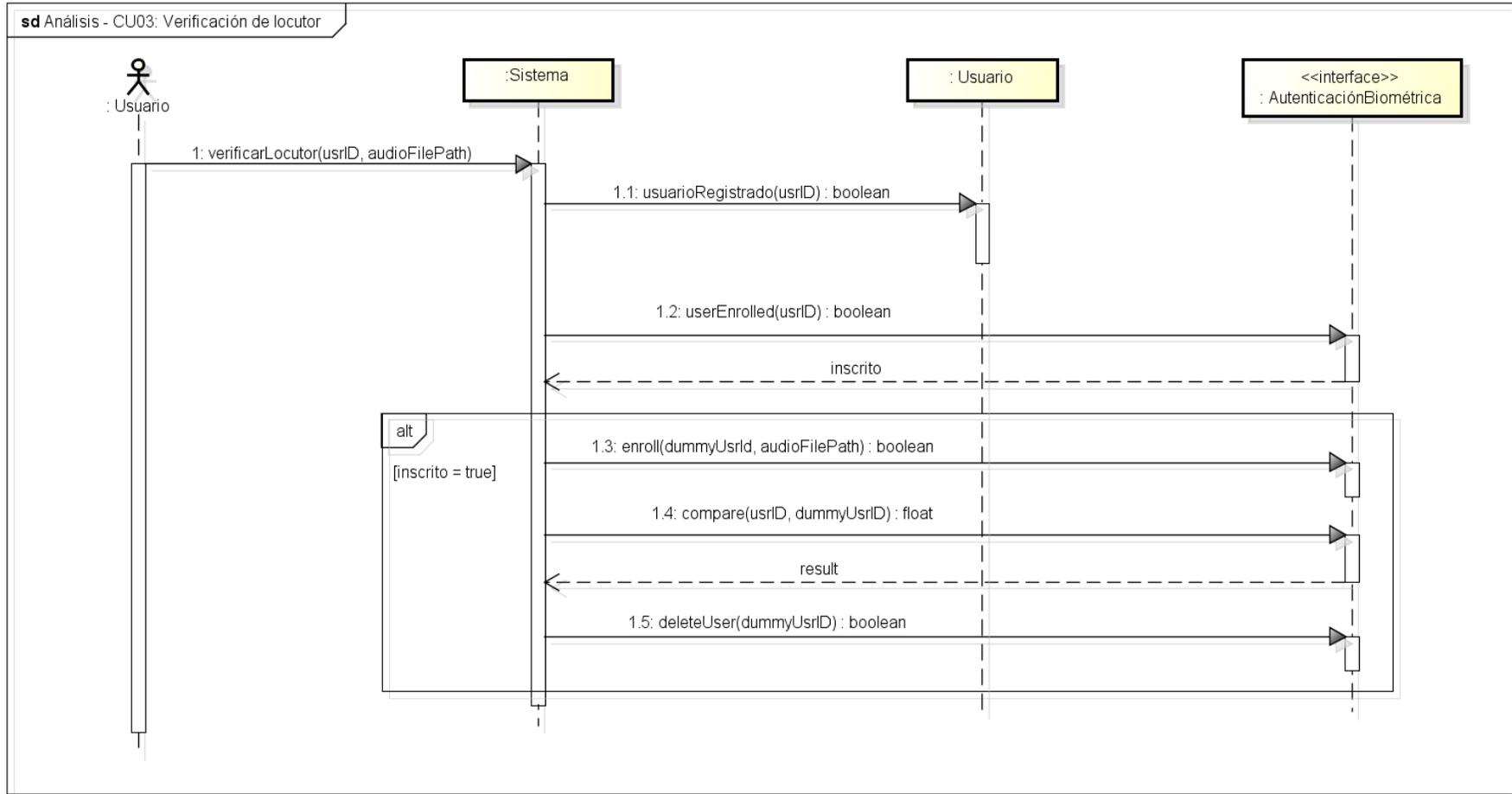
4.6.2. CU02 - Inscripción en el sistema

Figura 4.4.: Diagrama de secuencia CU02 - Inscripción en el sistema



4.6.3. CU03 - Verificación de locutor

Figura 4.5.: Diagrama de secuencia CU03 - Verificación de locutor



Capítulo 5.

Diseño

5.1. Introducción

Durante el siguiente capítulo se desarrollan las decisiones de diseño tomadas para la realización de la aplicación web, la aplicación Android y las tecnologías de servidor que las den soporte.

El presente capítulo, al igual que el análisis, sigue, aunque no estrictamente el modelo proporcionado por OpenUP. De nuevo, los diagramas realizados se crean utilizando la herramienta Astah Professional 6.8.0.

5.2. Objetivos arquitectónicos y filosofía

Este proyecto será utilizado por el departamento de investigación que lo ha propuesto, por lo que se debe conseguir un desarrollo con una flexibilidad y calidad suficiente para las necesidades particulares que un organismo de este tipo puede necesitar. Se tendrán en cuenta por tanto varios objetivos.

5.2.1. Reutilización

Habrán otros proyectos, ya sean trabajos fin de grado o proyectos internos de investigación realizados por el departamento, que necesiten hacer uso de partes de este desarrollo. Se debe, por lo tanto, asegurar la independencia de cada uno de los componentes que se detecten, haciendo que se puedan reutilizar de manera sencilla en proyectos totalmente distintos.

5.2.2. Extensibilidad

Para este proyecto se han definido una serie de aplicaciones a desarrollar (una aplicación web y una aplicación Android), y un sistema de autenticación biométrica que las de soporte (Alize).

No obstante, la propia naturaleza de este proyecto de investigación hace que estos límites, pertenecientes a este proyecto, puedan ser profundamente ampliados en otro. Debemos, por tanto, dar la posibilidad, en futuros desarrollos, de añadir componentes sin hacer grandes modificaciones en el sistema.

La adición, en concreto, de nuevos front-ends (aplicaciones móviles para otras plataformas, software de escritorio, ...), o nuevos sistemas de autenticación biométrica, así como de diferentes opciones para la ejecución de estos, debe ser muy sencilla.

5.2.3. Configurabilidad

Teniendo en cuenta que la verificación de locutor es un proceso que depende de un gran número de parámetros de configuración, debemos desarrollar un sistema en el que los cambios en estos parámetros sean fáciles de implementar.

En concreto, los valores de configuración no deben estar dentro del código fuente, sino en ficheros auxiliares de configuración, fácilmente accesibles y comprensibles.

5.2.4. Portabilidad

Las circunstancias tecnológicas de los departamentos de investigación son muy cambiantes. El hecho de realizar un proyecto que depende exclusivamente de estar instalado en un sistema operativo concreto o, aún más restrictivo, en una versión concreta de este, hace que los esfuerzos en lograr una gran reusabilidad sean en vano.

Es por esto que las tecnologías utilizadas no deben funcionar de forma exclusiva en una tecnología subyacente concreta. Es decir, debe haber opciones, tanto para usuarios como para administradores, de usar o desplegar este sistema sin importar el tipo de dispositivo o sistema operativo que utilicen.

5.3. Arquitectura del sistema

5.3.1. Visión global

Hay varios aspectos que definen claramente la arquitectura de este proyecto, teniendo en cuenta también los objetivos y filosofía mencionados en la sección 5.2.

Por un lado, se pretende desarrollar dos aplicaciones para plataformas distintas que accedan a una misma funcionalidad. Este aspecto define de manera clave nuestra arquitectura, dado que no tiene sentido realizar dos aplicaciones completas. Se creará, por tanto, un tercer componente que se encargue de recibir las peticiones de inscripción y verificación de locutor desde las plataformas web y Android.

El acceso al componente de autenticación biométrica se realizará desde dos plataformas completamente diferentes, lo cual nos inclina a utilizar una forma de comunicación entre componentes que sea soportada de manera masiva. Esta forma de comunicación será el protocolo HTTP, ya que tanto la aplicación web (por descontento) como la aplicación Android proporcionan grandes facilidades para generar peticiones y recibir respuestas usando este protocolo.

Para que podamos realizar las peticiones adecuadas al software de autenticación biométrica debemos, no obstante, diseñar una forma de comunicación sobre HTTP que nos de la expresividad suficiente. Así, llegamos a la conclusión de que la creación de una API Rest en el lado del servidor es la opción más adecuada.

Finalmente, será necesario el desarrollo de un componente más. Como se mencionó en el capítulo dos, el sistema de autenticación biométrica que se va a utilizar es Alize. Al estar desarrollado en C++, necesitamos un componente, incluido en el servidor, que envíe las peticiones a los diferentes comandos de Alize y traduzca las respuestas. Será realizado, también, en PHP.

Daremos un nombre a cada uno de estos componentes, de forma que sea más sencillo identificarlos en el resto del documento:

- **BioVoiceWeb:** Front-end web para el acceso a las capacidades de autenticación biométrica del sistema.
- **BioVoiceApp:** Aplicación android para el acceso a las capacidades de autenticación biométrica del sistema.
- **BioVoiceAPI:** API Restful que permite a los Front-ends interactuar con el sistema de autenticación biométrica.

- **AlizePHP:** Componente encargado de enviar las peticiones pertinentes a Alize, el software de autenticación biométrica, y traducir sus respuestas de una forma comprensible para la API.

Una vez definidos los componentes del sistema debemos prestar atención a las tecnologías a utilizar. Para el desarrollo de la aplicación Android utilizaremos el SDK que nos proporciona Google para realizar aplicaciones nativas para esta plataforma. El resto de opciones disponibles, aunque cuentan con la ventaja de poder generar ejecutables para varias plataformas, son menos maduras.

Para el resto de tecnologías los principales criterios a tener en cuenta, tratándose este proyecto de un trabajo Fin de Grado son los siguientes:

1. **Facilidad de uso:** Para poder realizar el proyecto satisfactoriamente en la ventana temporal requerida.
2. **Amplitud de documentación:** Que permita que el alumno pueda resolver las dudas que le surjan utilizando los recursos a su disposición.
3. **Coste:** El software usado debe ser gratuito u obtenido de una forma gratuita.

Atendiendo a estas características, parece adecuado inclinarse por sistemas gratuitos, de código abierto, que sean lo suficientemente populares para tener el apoyo de una amplia comunidad y con una documentación completa.

Para el servidor, que incluye la API Rest y el acceso al software de autenticación biométrica, se utilizará, por tanto, una pila LAMP (Sistema operativo **L**inux, servidor web **A**pache, base de datos **M**ySQL y lenguaje de programación **P**HP), una solución ámpliamente probada con éxito en proyectos de este tipo.

La aplicación web tendrá como tecnología de servidor, también, PHP. Para la parte de cliente se utilizará HTML5, CSS y JavaScript. Uno de los aspectos clave de este proyecto es que se debe poder recoger el audio en un navegador sin la necesidad de instalar complementos de terceros, por lo que el navegador utilizado debe implementar la tecnología WebRTC, como se ha mencionado en los requisitos no funcionales del proyecto reflejados en el apartado 4.2.3.

En el diagrama de despliegue de la figura 5.1 se representan estas características. En este caso, se utilizan tres equipos, el dispositivo móvil, un dispositivo con un navegador y el servidor, que incluye la aplicación web, la API Rest, el componente de acceso a Alize y el propio software de autenticación biométrica, así como el sistema gestor de bases de datos.

En la figura 5.2, podemos observar el diagrama de componentes, con las distintas partes intercambiables del sistema y las interfaces requeridas y proveídas por cada una de ellas.

Figura 5.1.: Diagrama de despliegue

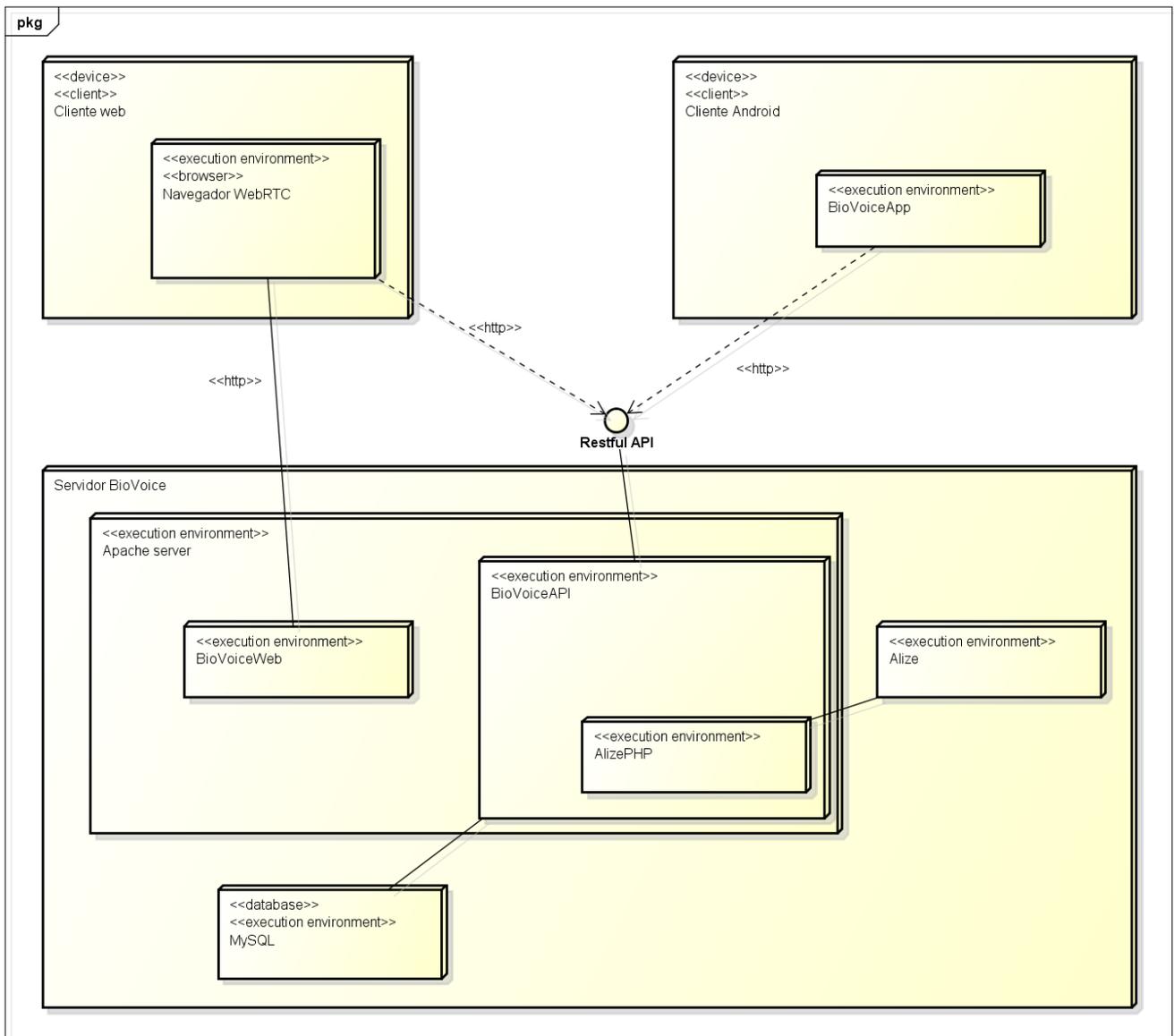
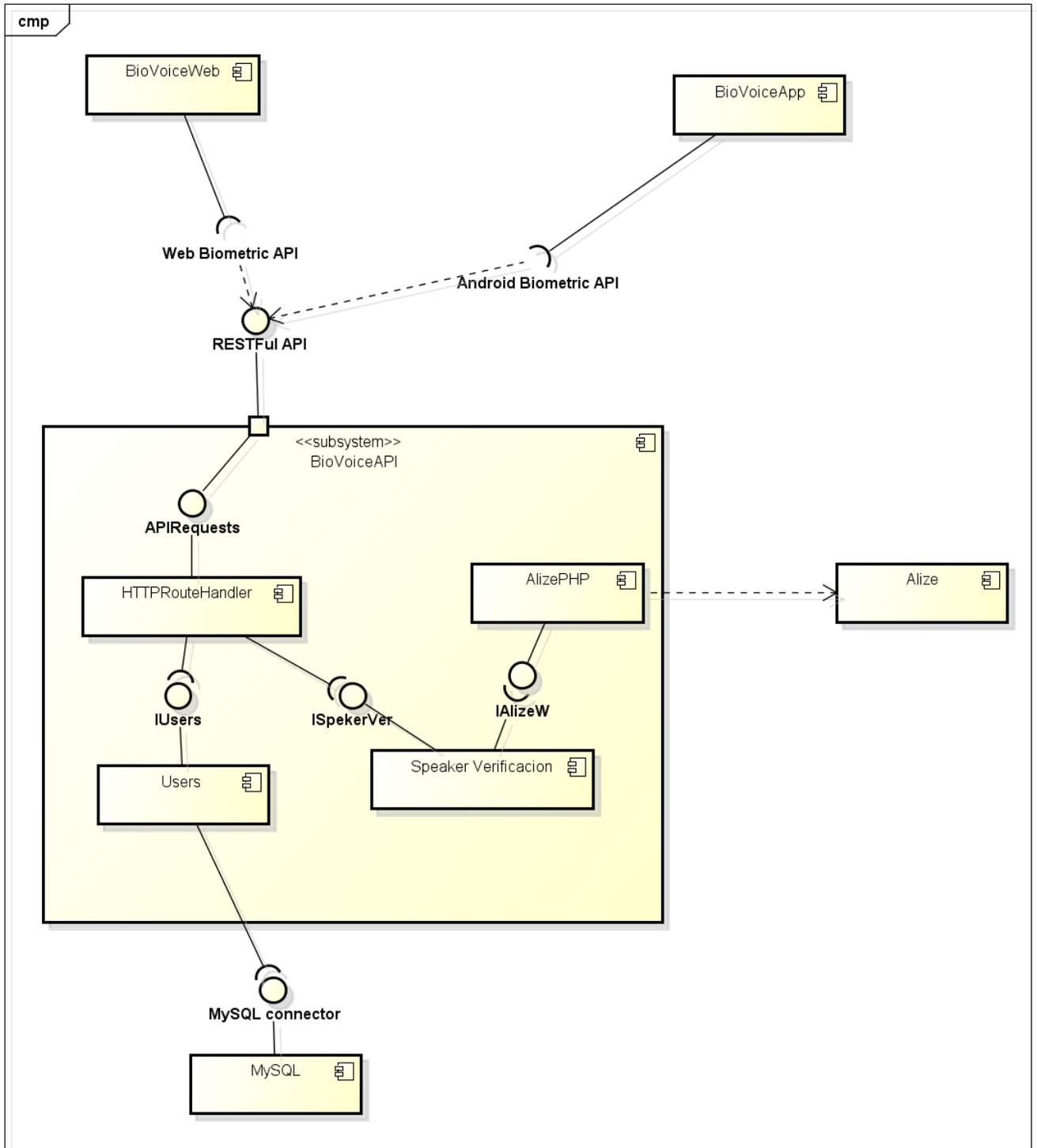


Figura 5.2.: Diagrama de componentes



Se sigue una arquitectura basada en capas, habitual en las aplicaciones web, pero con las particularidades inherentes al diseño de este proyecto. Por un lado, la capa de presentación tiene dos componentes, la aplicación web y la aplicación Android. La lógica de negocio se encuentra en los controladores y los manejadores de las rutas http. La lógica de dominio, por último, está también dividida entre los datos almacenados en el SGBD y los datos que maneja Alize, principalmente ficheros binarios.

Por otra parte, los componentes de front-end y la API siguen el patrón MVC, aunque también con diferencias. Los componentes front-end no disponen de modelo interno, sino que la API es quien actúa como su modelo. De la misma manera, la API no tiene vistas, ya que delega la función de mostrar los datos en los front-ends que hagan uso de ella. Esto hace que el patrón MVC sea global, siendo las Vistas los front-ends, estando los controladores diseminados entre la API y las aplicaciones, y contando con un modelo en la API.

La figura 5.3 muestra esta arquitectura de manera más detallada.

5.3.2. Diseño de la arquitectura

Descomposición en subsistemas

Es una buena práctica, a la hora de diseñar un proyecto, obtener la descomposición en subsistemas partiendo de los casos de uso. El proyecto que nos ocupa no tiene un gran número de casos de uso, por lo que tendremos que refinar un poco más esta descomposición basándonos en la separación en componentes realizada en la sección 5.3.1.

El primero de los casos de uso, “CU01 - Registro en el sistema”, da una clara idea de que será necesario un subsistema de gestión de usuarios.

Los dos siguientes, “CU02 - Inscripción en el sistema” y “CU03 - Verificación de locutor”, nos plantean la creación de un subsistema de autenticación biométrica.

Con estos dos subsistemas únicamente, como ya se ha advertido, no es posible definir la partición lógica del sistema. En realidad, estos dos subsistemas estarán incluidos BioVoiceAPI, siendo este el metapaquete que comprende a ambos, y dando servicio a los dos paquetes de front-end.

Obtenemos, por tanto, una división en subsistemas con múltiples similitudes con la partición física vista en el diagrama de despliegue. El diagrama de paquetes resultante se muestra en la figura 5.4.

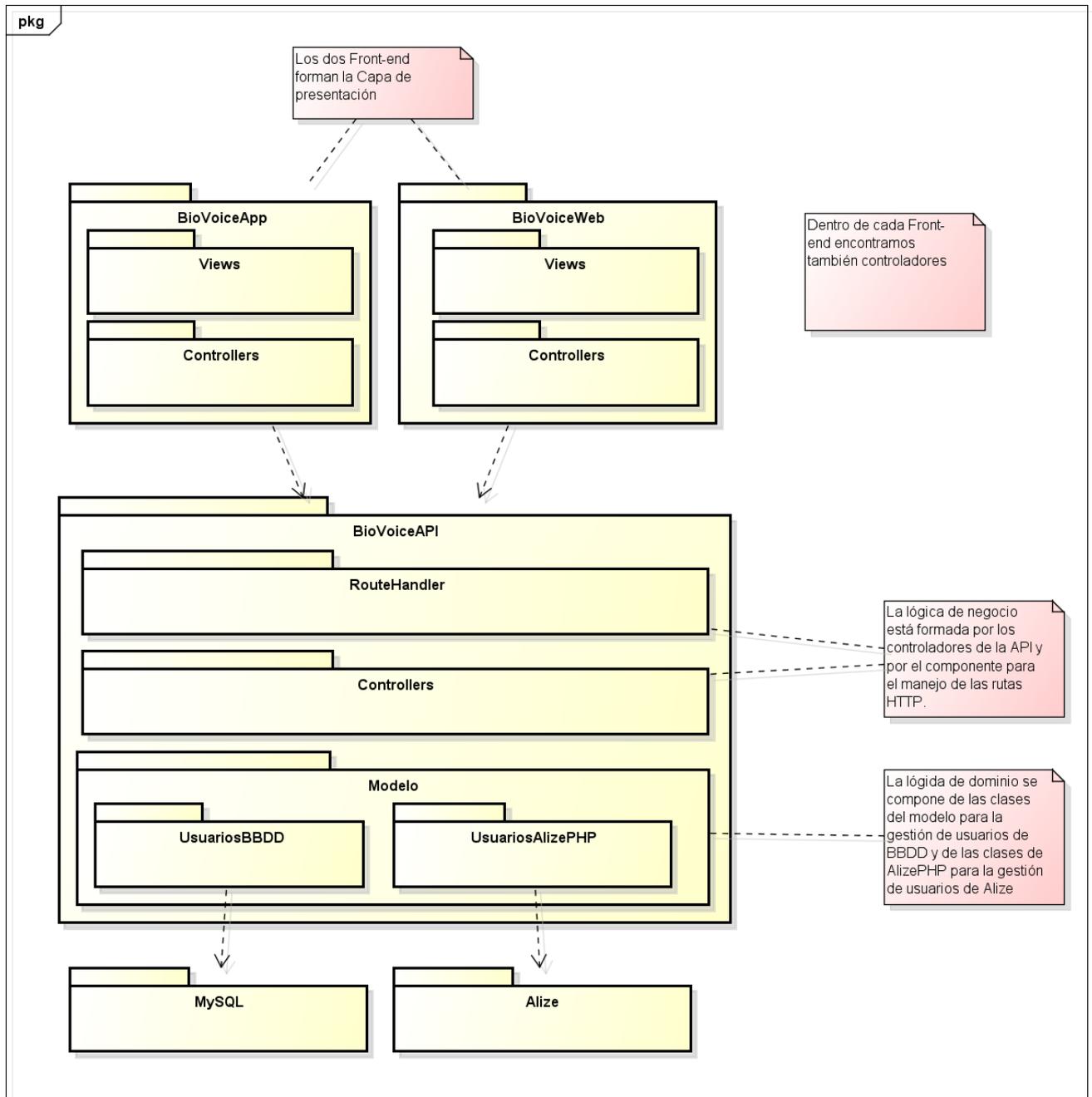
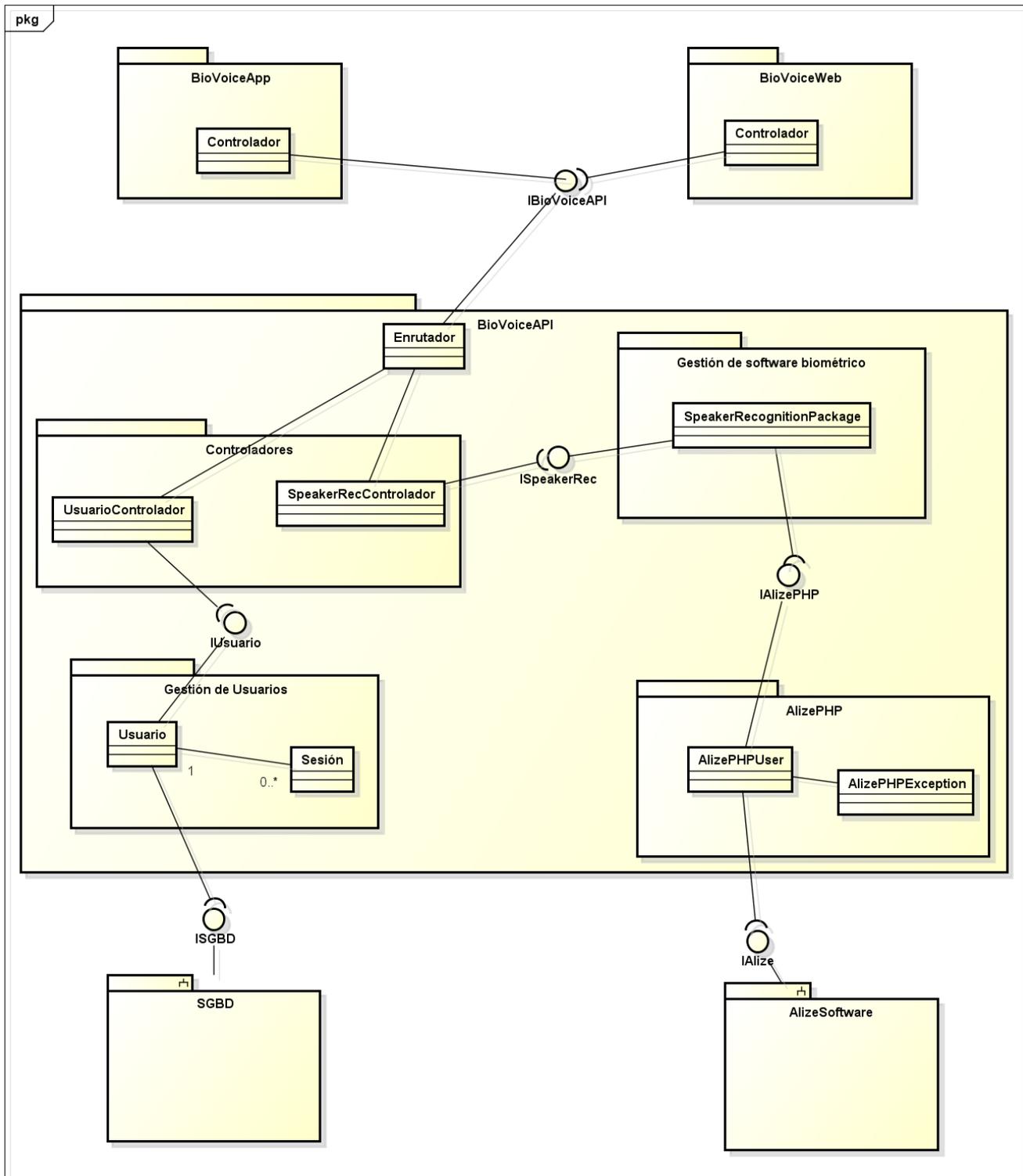


Figura 5.3.: Diagrama de capas

Figura 5.4.: Diagrama de paquetes



Gestión de la persistencia

Debemos dividir la gestión de la persistencia en dos partes.

La primera de ellas trata sobre la persistencia de los datos de los usuarios registrados, a almacenar en la base de datos MySQL. Se utilizará, en este caso, el patrón Active Record, debido a su sencillez de implementación y uso, teniendo en cuenta que el modelo de dominio que manejamos es tremendamente simple.

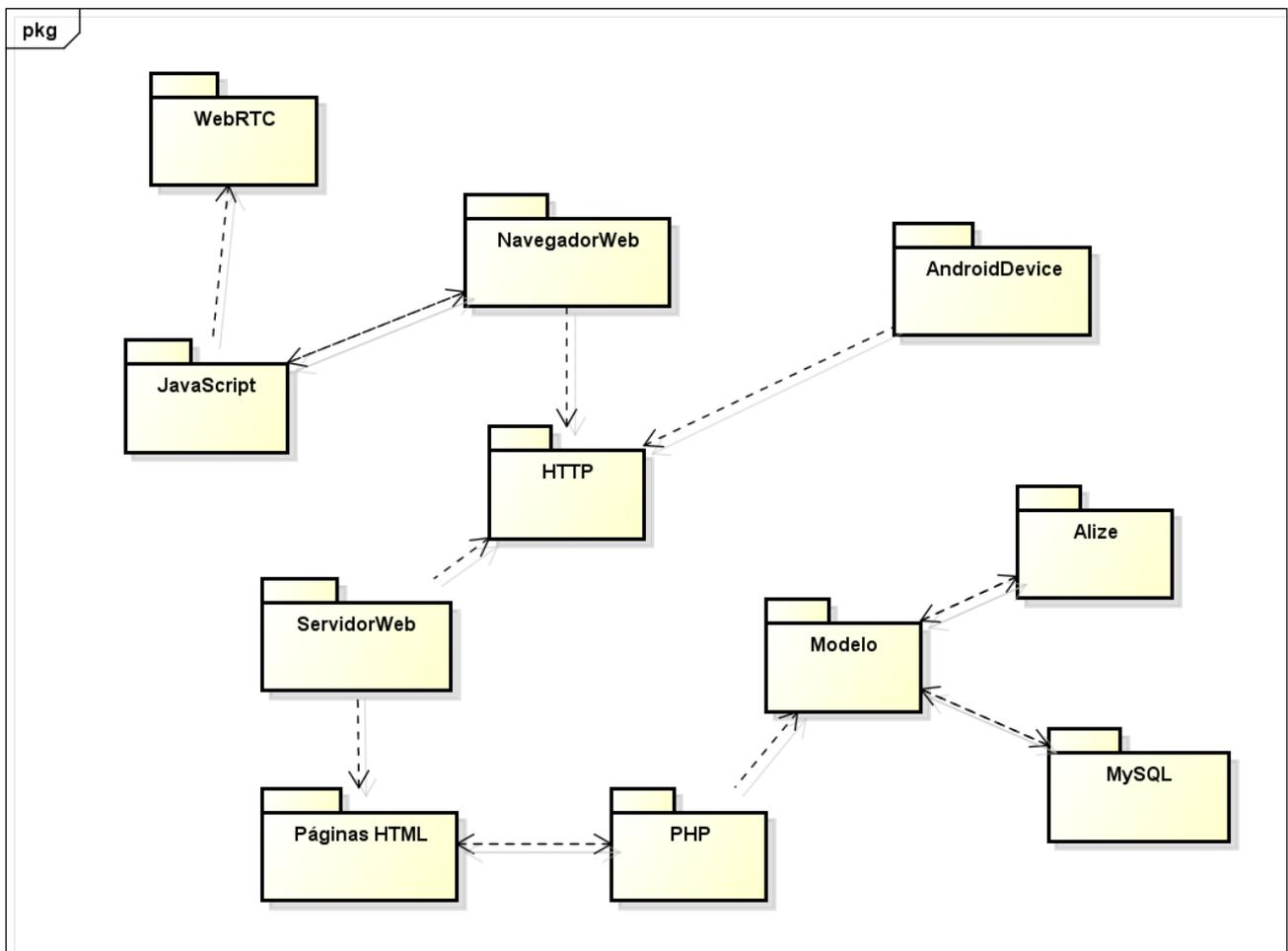
Por otra parte, debemos tener en cuenta la gestión de la persistencia de los usuarios inscritos en el sistema de autenticación biométrica. Aunque este aspecto está fuera de los límites del proyecto, por ser gestionado por el software Alize, sí es importante reconocer el modo en que se almacenan los datos de los locutores.

Alize tiene un sistema propio de archivos encargado de almacenar los diferentes ficheros binarios y de texto que utiliza en cada una de las etapas de generación del modelo de un locutor. De la creación y el borrado de estos ficheros se encargará Alize, pero de indicar la ubicación de cada uno de ellos se deberá encargar el componente que desarrollemos en este proyecto: AlizePHP.

Vista lógica

Por último, se presenta la vista lógica del sistema en la figura 5.5, donde se muestran los principales componentes del diseño y las relaciones entre los mismos.

Figura 5.5.: Vista lógica del sistema



5.4. Patrones

Se detallan a continuación algunos de los patrones usados para alcanzar los objetivos arquitectónicos descritos en la sección 5.2, teniendo en cuenta la arquitectura desarrollada durante la sección 5.3.

5.4.1. Adaptador (Wrapper)

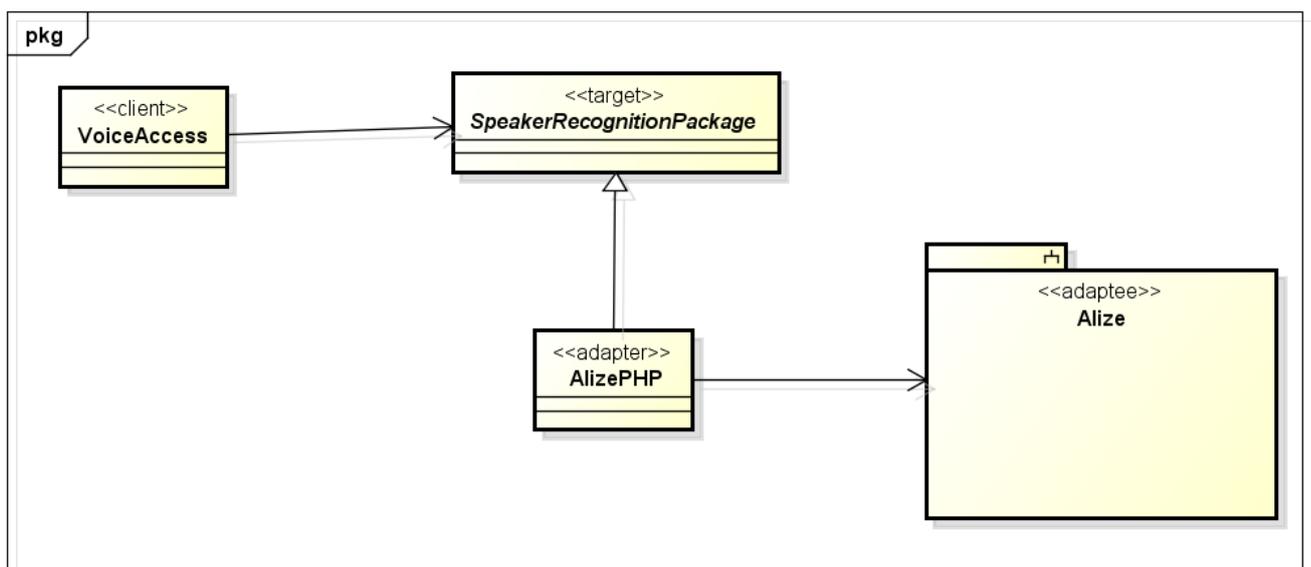
Este patrón se ha utilizado para desarrollar el componente denominado AlizePHP. Su propósito se define como el de convertir la interfaz de una clase a otra interfaz, esperada por el cliente. [GHJV95]

La utilización de este patrón permite, por tanto, utilizar el software Alize, basado en comandos y ficheros de configuración, desde cualquier aplicación escrita en PHP, utilizando un enfoque orientado a objetos y abstrayendo a la aplicación PHP de toda la complejidad con la que cuenta el software Alize, en lo que a gestión de ficheros y binarios se refiere.

Con este patrón conseguimos el objetivo de reusabilidad, dado que permitimos que Alize se pueda incluir fácilmente en otros proyectos web escritos en PHP.

En la figura 5.6 se muestra cómo se ha implementado este patrón en el proyecto.

Figura 5.6.: Patrón adaptador



5.4.2. Fachada

Un sistema de autenticación biométrica puede (y tiene que) realizar una gran cantidad de acciones para realizar lo que, desde el punto de vista de nuestro proyecto, es una acción atómica: la inscripción o la verificación de un locutor.

Debido a esto, debemos mejorar la arquitectura diseñada con el patrón adaptador, vista en la sección 5.4.1, ya que, tal y como habíamos diseñado la interacción entre el cliente y Alize, debíamos hacer un adaptador que cumpliera con las normas que le imponía la interfaz “SpeakerRecognition-Package”.

Solucionaremos esto creando una fachada para AlizePHP. El patrón fachada se utiliza para proveer de una interfaz unificada a una serie de interfaces en un subsistema [GHJV95]. Aunque nuestro subsistema, AlizePHP, no tiene varias interfaces, la complejidad de su única clase hace aconsejable este enfoque, así como la necesidad, una vez más, de mantener el objetivo de reusabilidad y, en este caso, también el de extensibilidad.

Podemos observar la adición de este patrón a la arquitectura anterior en la figura 5.7.

5.4.3. Método factoría

Como se ha desarrollado en la sección 5.2, una de las características deseables para nuestro sistema, dentro del objetivo de extensibilidad, es la de poder añadir la funcionalidad proporcionada por otro sistema de autenticación biométrica distinto de Alize.

Para simplificar este proceso, podemos hacer uso del patrón factoría a la hora de instanciar el software biométrico. Este patrón define una interfaz para la creación de un objeto, pero deja que las clases descendientes decidan qué clases instanciar, es decir el método factoría permite a una clase posponer la instanciación a las subclasses [GHJV95].

Aplicado a nuestro diseño, esto nos permite que, añadir nuevos sistemas de reconocimiento de locutor, no tenga mayor impacto sobre el código de nuestra API, y por extensión de nuestro proyecto. Sólo será necesario crear un nuevo constructor concreto y un nuevo producto concreto para que el nuevo sistema se pueda utilizar.

La figura 5.8 muestra este patrón aplicado a la unión de BoiVoiceAPI y AlizePHP.

Figura 5.7.: Patrón fachada

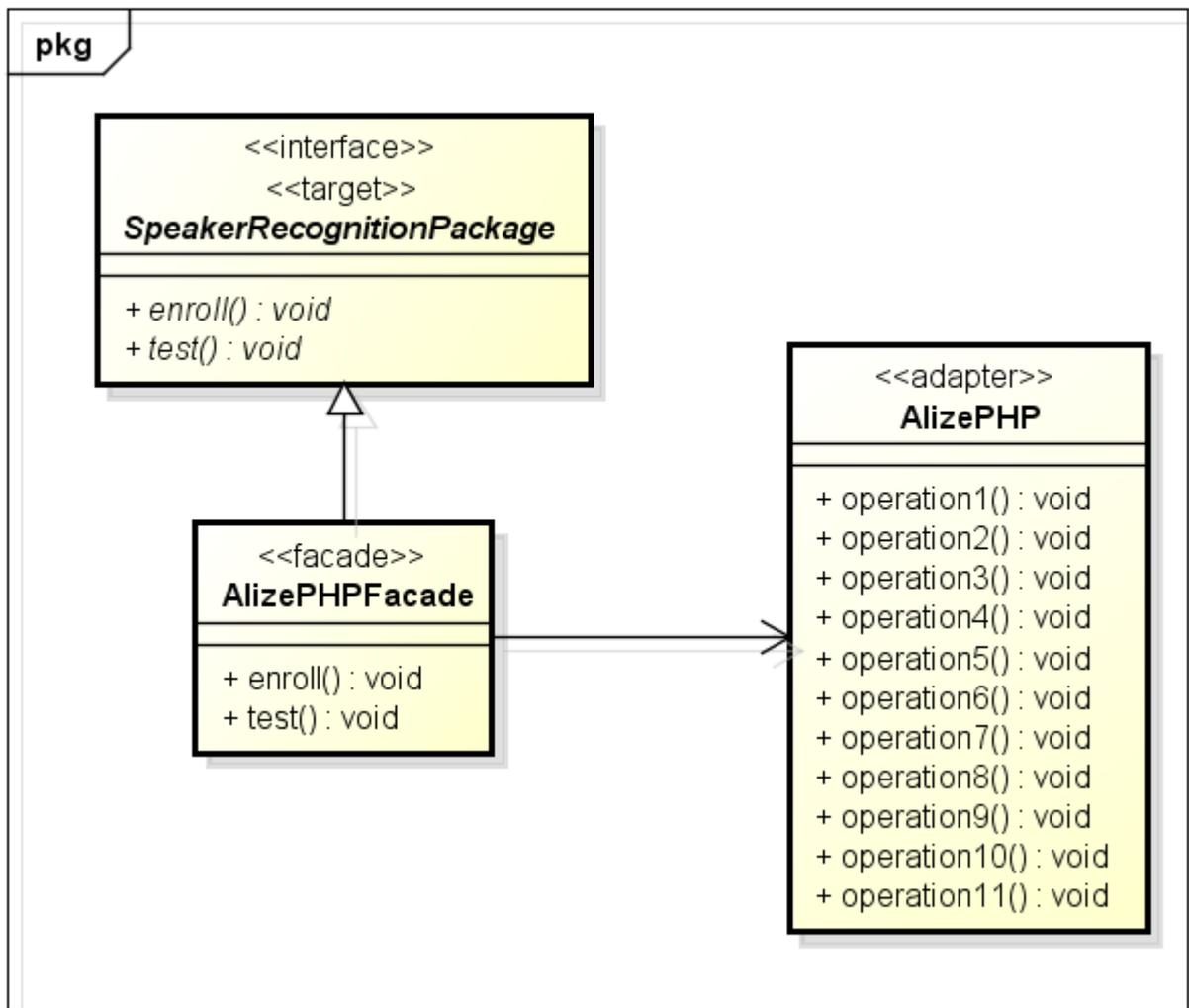
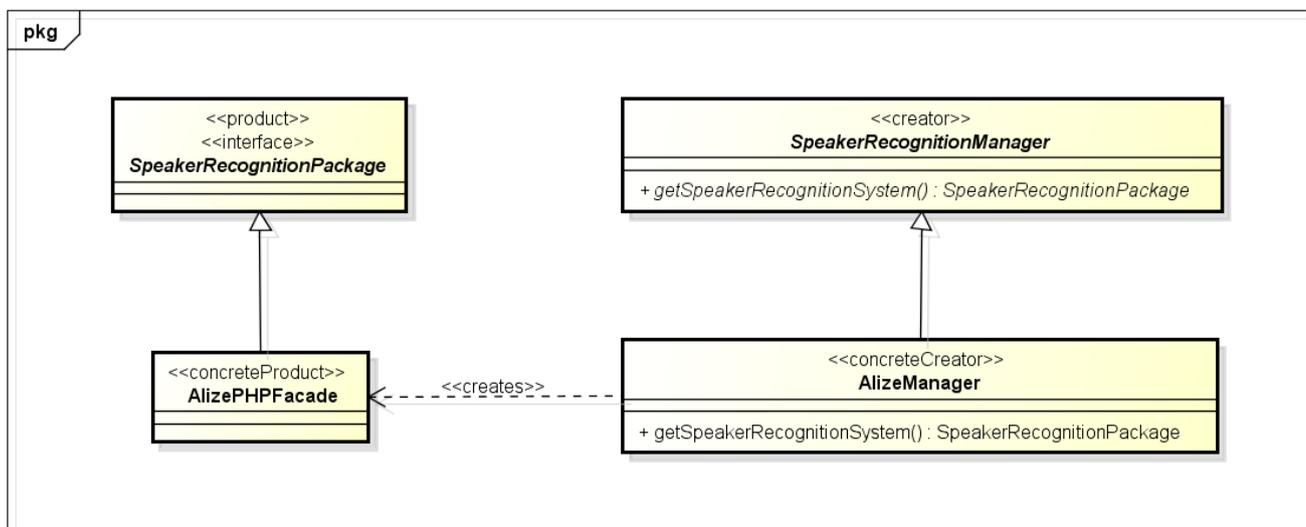


Figura 5.8.: Patrón método factoría



5.5. Casos de uso de diseño

5.5.1. CU01 - Registro en el sistema

Descripción

Un usuario debe registrarse en el sistema para poder realizar la inscripción de su voz.

Actores

- Usuarios

Precondiciones

No hay precondiciones

Flujo normal de eventos

1. El caso de uso comienza cuando el miembro accede al sistema a través de su navegador o aplicación Android y pulsa en la opción “Registro”.
2. El sistema pide, a través de un formulario, los datos que figuran en el RI01 de la sección 4.2.4.
3. El miembro rellena el formuarli con los datos solicitados y presiona en el botón “Registro” para efectuar su registro.
4. El sistema comprueba que todos los datos introducidos son válidos y almacena los nuevos datos del usuario en la base de datos.
5. El sistema almacena los nuevos datos del usuario en la base de datos.
6. El sistema muestra por pantalla al usuario un mensaje de confirmación de que el proceso se ha realizado correctamente.
7. El caso de uso termina.

Flujo alternativo

textbfFallo de validez de datos

Si en el paso 4 el sistema comprueba que al menos uno de los datos no es válido.

1. El sistema muestra esta información al usuario.
2. El caso de uso finaliza con error.

Si en el paso 5 el sistema no consigue crear el usuario.

1. El sistema muestra esta información al usuario.
2. El caso de uso finaliza con error.

Escenarios clave

Roberto, una persona interesada en los sistemas biométricos, accede a este sistema. Como quiere probar qué tal funciona inscribiendo su voz y haciendo diferentes test, decide registrarse. El sistema le pide los datos necesarios para hacer la inscripción, Roberto los proporciona y recibe un mensaje de confirmación indicándole que se ha registrado correctamente. Desde este momento, Roberto puede inscribir su voz en el sistema.

Postcondiciones

Finalización satisfactoria

Los datos del nuevo usuario quedan guardados en la base de datos.

Finalización con errores

Los datos del usuario no se guardarán en la base de datos, y se generará un error en el fichero de log correspondiente.

5.5.2. CU02 - Inscripción en el sistema

Descripción

El usuario desea inscribir su voz en el sistema de autenticación biométrica.

Actores

- Usuario registrado

Precondiciones

- El usuario debe estar registrado en el sistema.

Flujo normal de eventos

1. El caso de uso comienza cuando el usuario accede al sistema y pulsa en la opción “Inscripción”.
2. El sistema pide el nombre de usuario del que se quiere realizar la inscripción.
3. El usuario introduce el nombre de usuario del usuario registrado del que se quiere hacer la inscripción.
4. El sistema comprueba la validez del nombre de usuario.
5. El sistema muestra una pantalla en la que el usuario pueda usar el micrófono de su equipo para grabar su voz.
6. El usuario pulsa en la opción de grabar y graba su voz, cuando ha terminado pulsa en la opción de parar y envía el audio al sistema.
7. El sistema recoge la muestra de audio e indica al sistema de autenticación biométrica que debe inscribir al usuario indicado con el modelo del audio que se le envía.
8. El sistema muestra un mensaje al usuario indicándole que el proceso de inscripción se ha realizado correctamente.
9. El caso de uso termina.

Flujo alternativo

Nombre de usuario no válido

Si en el paso 4 el sistema comprueba que no hay ningún usuario registrado con ese nombre.

1. El sistema informa al usuario mediante un mensaje de error.
2. El caso de uso finaliza.

Usuario ya inscrito

Si en el paso 4 el sistema comprueba que el usuario ya está inscrito.

1. El sistema informa al usuario usuario de este hecho.
2. El caso de uso continúa en el paso 5.

Error al procesar audio

Si en el paso 7 el software de autenticación biométrica no consigue completar el procesamiento del audio.

1. El sistema informa al usuario mediante un mensaje de error.
2. El caso de uso finaliza.

Cancelación

En cualquiera de los pasos el usuario indica que quiere cancelar el caso de uso.

1. El caso de uso finaliza.

Escenarios clave

Un usuario registrado, David, decide probar qué tal funciona la verificación de locutor del sistema. Para ello, indica al sistema que quiere realizar la inscripción proporcionando sus credenciales, el sistema le pide un fichero de audio con su voz y David lo envía. El sistema recoge el fichero de audio y se lo pasa al software de autenticación biométrica para que procese la muestra de voz de David. Se genera su modelo correctamente, por lo que el sistema muestra a David un mensaje de confirmación indicándole el éxito de su inscripción.

Postcondiciones

Finalización satisfactoria

El software de autenticación biométrica generará los datos oportunos para almacenar las características de la voz del usuario inscrito. Se informará por pantalla al usuario del éxito del proceso.

Finalización con errores

Se mostrará un mensaje de error y se almacenarán los datos de la ejecución en el fichero de log correspondiente.

5.5.3. CU03 - Verificación de locutor

Descripción

Un usuario realiza una verificación de locutor.

Actores

- Usuario
- Usuario registrado

Precondiciones

- El usuario debe estar autenticado en el sistema como usuario final o usuario desarrollador.

Flujo normal de eventos

1. El caso de uso comienza cuando el usuario accede al sistema y pulsa en “Verificación de locutor”.
2. El sistema solicita el nombre de usuario del usuario contra el que se va a realizar la verificación.
3. El usuario proporciona la el nombre de usuario del usuario contra el que se quiere realizar la verificación.
4. El sistema comprueba que la identidad del usuario es válida.
5. El sistema muestra una pantalla en la que el usuario pueda usar el micrófono de su equipo para grabar su voz.
6. El usuario pulsa en la opción de grabar y graba su voz, cuando ha terminado pulsa en la opción de parar y envía el audio al sistema.
7. El sistema recoge la muestra de audio e indica al sistema de autenticación biométrica que debe generar el modelo de la muestra de uadio indicada y compararlo, posteriormente, con el del usuario contra el que se quiere hacer la verificación.
8. El sistema muestra un mensaje al usuario indicándole que el proceso de verificación se ha realizado correctamente, y el valor final que ha tenido esta.

9. El caso de uso termina.

Flujo alternativo

Usuario no registrado

En el paso 4 el usuario introduce la identificación de un usuario no registrado.

1. El sistema indicará mostrará un mensaje informando de que el usuario indicado no existe.
2. El caso de uso termina.

Usuario no inscrito

En el paso 4 el usuario introduce la identificación de un usuario que aún no ha realizado la inscripción.

1. El sistema indicará mostrará un mensaje informando de que el usuario indicado no ha realizado la inscripción.
2. El caso de uso termina.

Error al procesar audio

Si en el paso 7 el software de autenticación biométrica no consigue completar el procesamiento del audio.

1. El sistema informa al usuario mediante un mensaje de error.
2. El caso de uso finaliza.

Cancelación

En cualquiera de los pasos el usuario indica que quiere cancelar el caso de uso.

1. El caso de uso finaliza.

Escenarios clave

Natalia accede al sistema. Hace unos días se registró y realizó la inscripción, así que se dispone a comprobar qué tal funciona la verificación de locutor. Introduce su identificador de usuario en el sistema, que le pide una muestra de audio con su voz. Natalia proporciona dicha muestra de audio, el software de autenticación biométrica la procesa y la compara con los datos que generó y almacenó en el momento en que natalia se inscribió, proporcionando un valor de la similitud entre ambas voces. Este valor se le muestra a Natalia.

A Laura siempre le han dicho que tiene una voz muy parecida a la de Natalia. Natalia le habla de este sistema y Laura decide probar si realmente su voz podría pasar por la de Natalia. Laura nunca se ha registrado en el sistema, pero accede al sistema, introduce el identificador de Natalia y proporciona una muestra de audio de su voz. El sistema la procesa y la compara con los datos que generó para la voz de Natalia cuando esta se inscribió, mostrando un valor de la similitud de las dos voces a Laura.

Postcondiciones

Finalización satisfactoria

El software de autenticación biométrica realizará la comparación entre los dos usuarios. Obtendrá el valor de la distancia entre ambos modelos y se lo mostrará por pantalla al usuario.

Finalización con errores

Se mostrará un mensaje de error y se almacenarán los datos de la ejecución en el fichero de log correspondiente.

5.6. Diagramas de clases y de secuencia

Se presentan, a continuación, las vistas lógicas del sistema más descriptivas, compuestas por el diagrama de clases (figura 5.9) y los diagramas de secuencia de los tres casos de uso (figuras 5.10, 5.11 y 5.12).

Los diagramas de secuencia están realizados tomando como front-end la aplicación web. Sin embargo, la combinación de vistas y controladores en la aplicación Android será muy similar, y el diagrama en este caso sufriría cambios mínimos.

Figura 5.9.: Diagrama de clases de diseño

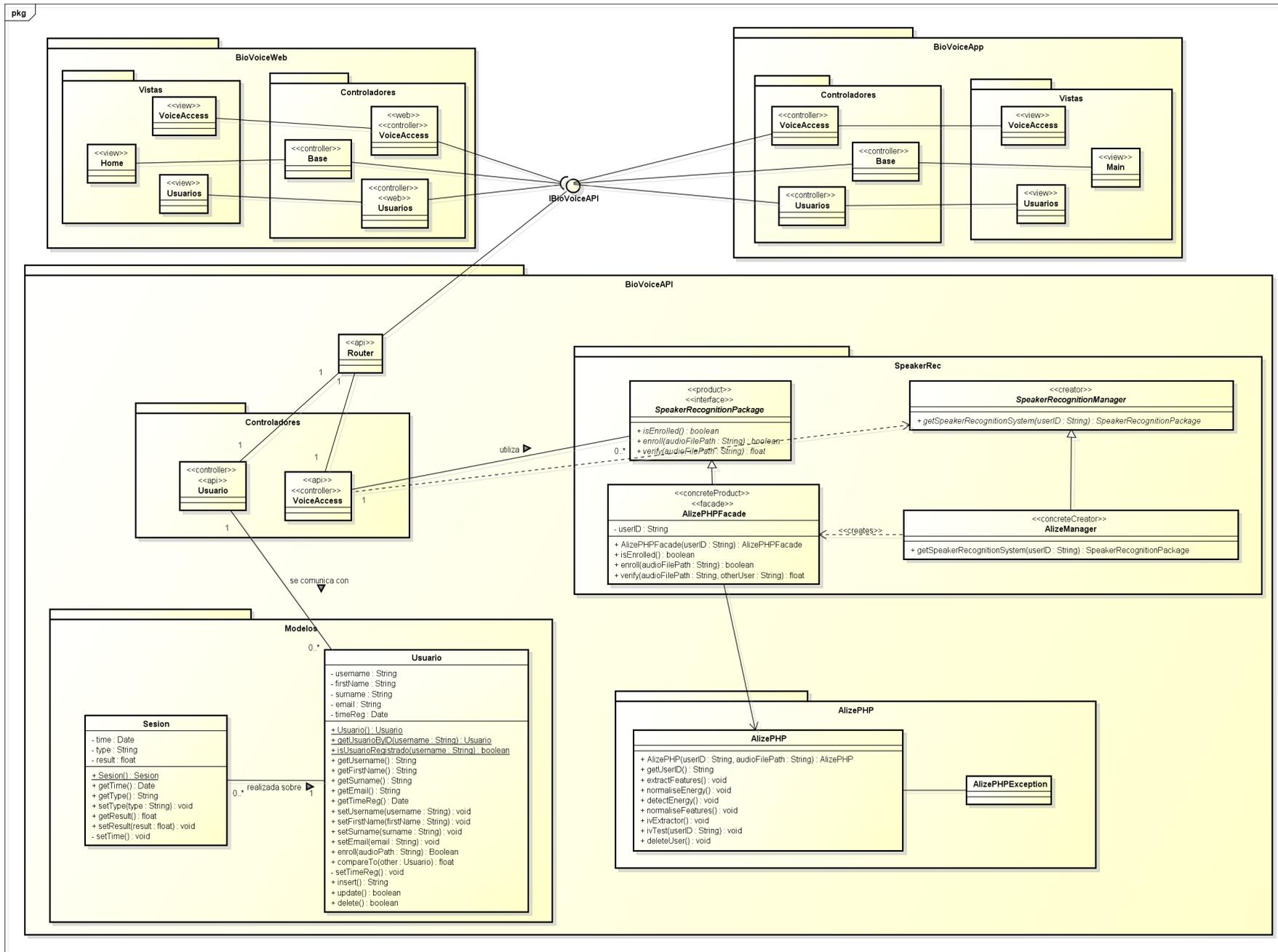


Figura 5.10.: CU01 - Registro en el sistema

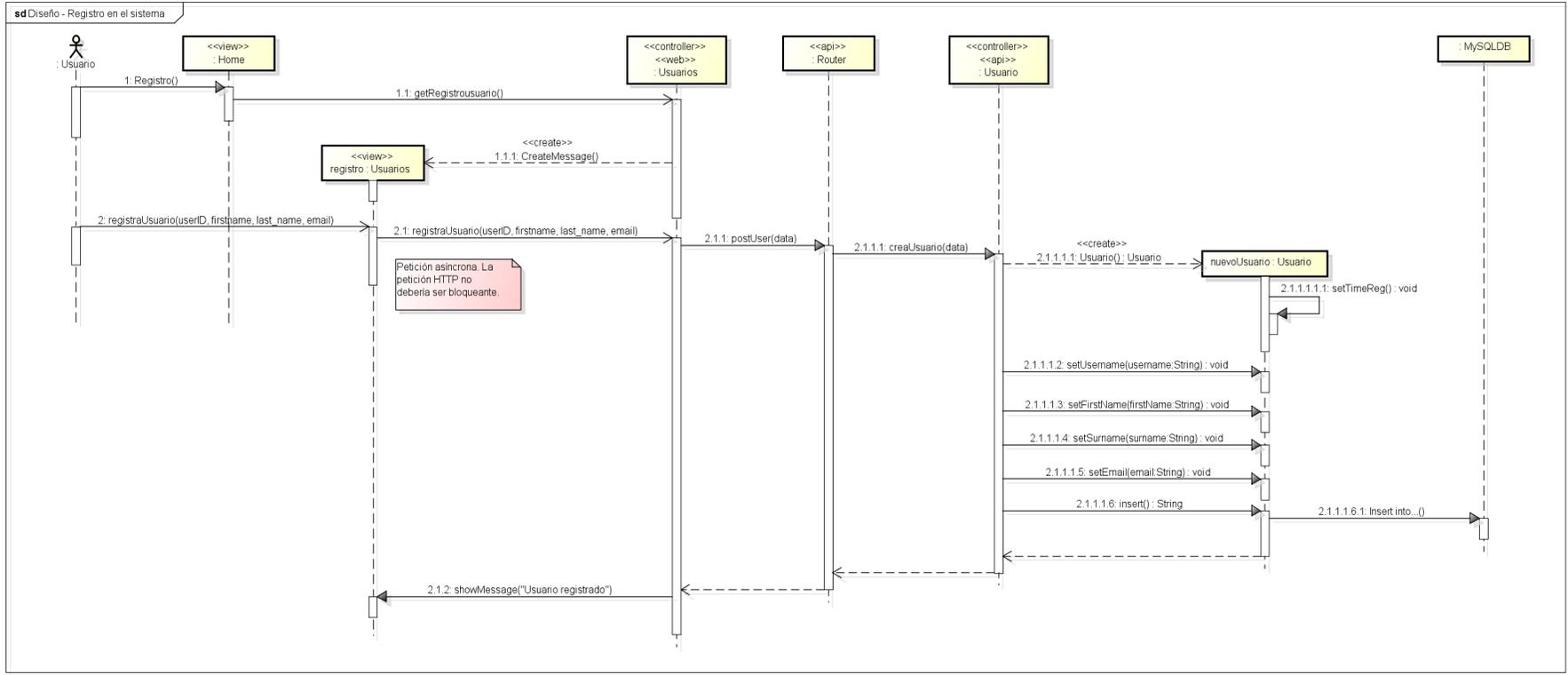


Figura 5.11.: CU02 - Inscripción en el sistema

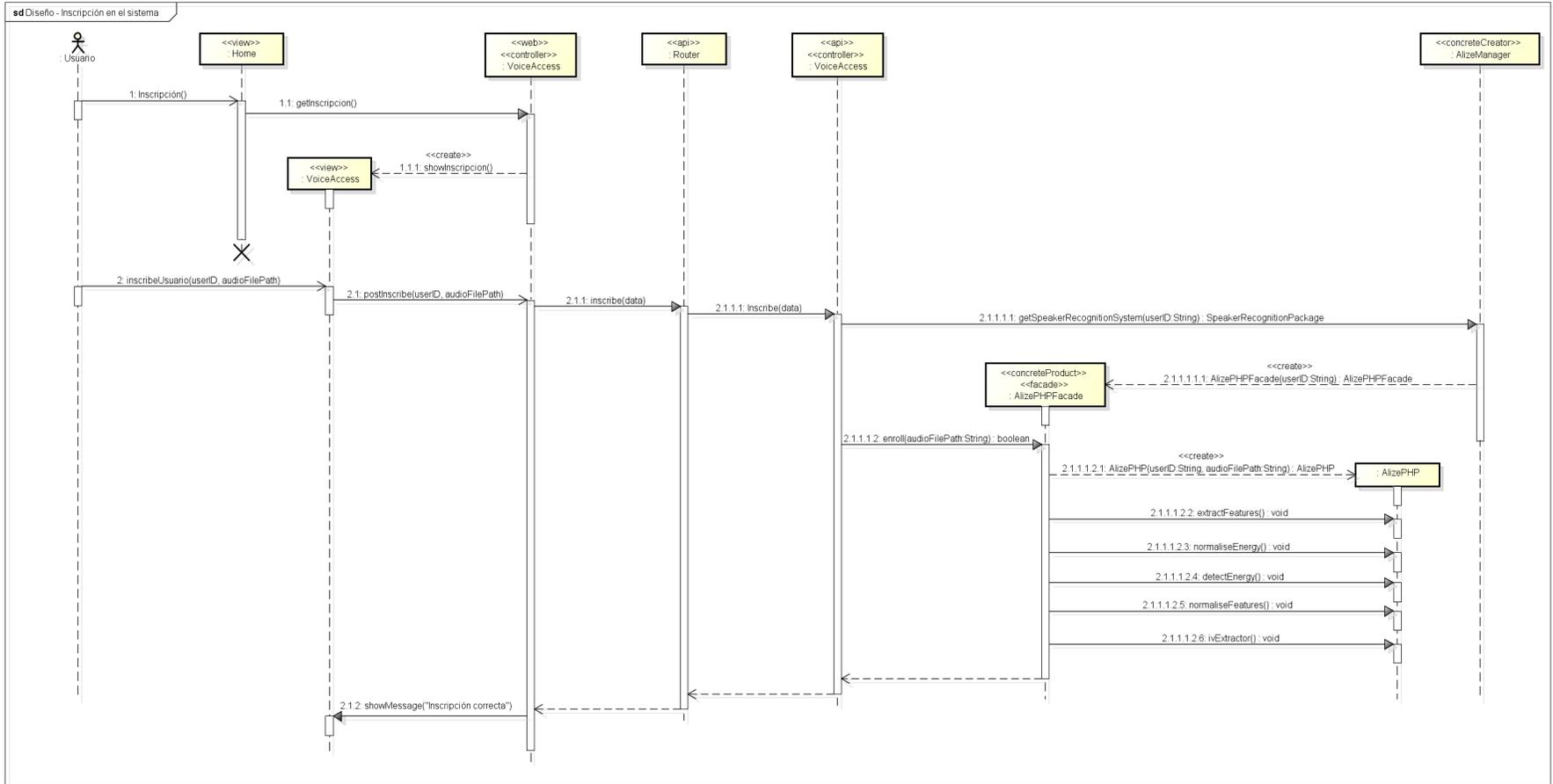
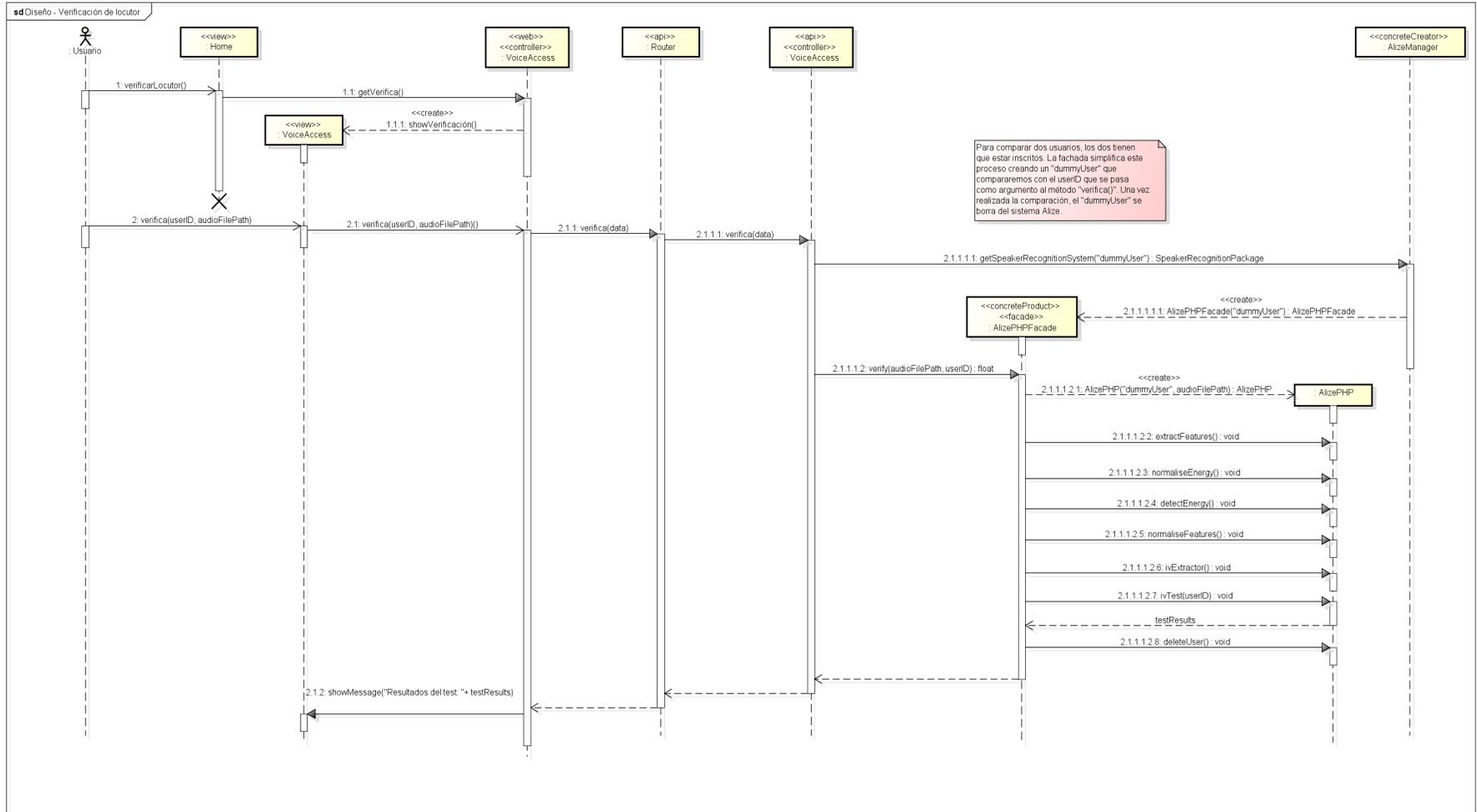


Figura 5.12.: CU03 - Verificación de locutor



Capítulo 6.

Implementación

6.1. Introducción

En este capítulo se detallan algunos de los puntos más relevantes de la implementación, relativos a las tecnologías utilizadas y los motivos fundamentales para su elección. La documentación interna que acompaña al código fuente complementa lo aquí expuesto en lo concerniente a los detalles de más bajo nivel.

6.2. Elementos relevantes en el desarrollo

El objetivo de este capítulo es mencionar aquellas herramientas, librerías o frameworks que han sido relevantes para el desarrollo del proyecto.

6.2.1. Servidor

Hemos elegido tecnologías para el desarrollo que permiten su despliegue en un servidor que tenga instalado cualquier tipo de sistema operativo: Apache, Alize y PHP pueden ser utilizados en Windows, Mac OS X o Linux. La elección del S.O., por tanto, no estará condicionada por el software, sino por la facilidad que este nos da a la hora de configurar los diferentes componentes que requiere el servidor y la familiaridad que tengamos con su uso.

Teniendo en cuenta estas métricas, el S.O. más adecuado para nuestros propósitos es Linux. Tanto por la familiaridad que tiene el alumno encargado del desarrollo con las acciones necesarias para desplegar un servidor en este sistema, como por la gran cantidad de posibilidades de configuración que ofrece. En concreto, se elige la distribución Debian 7.5, que tiene las características más actuales que se pueden encontrar en linux, aún manteniendo un conjunto de paquetes instalados mínimo.

6.2.2. IDEs y software de pruebas

El software base para el desarrollo ha consistido, principalmente, en dos IDEs. Para la programación en PHP se ha utilizado Eclipse con el plugin PDT (*PHP Development Tools*). Para la programación de la aplicación Android se utilizó también, en un principio, el IDE Eclipse con el plugin ADT (*Android Development Tools*), no obstante, viendo la complejidad que conllevaba gestionar las diferentes dependencias del proyecto BioVoiceApp en este software, se migró a Android Studio que, pese a ser una versión Beta, gestionó de forma más acertada la configuración de la app desarrollada.

Para la realización de las pruebas de la API REST ha sido clave el uso de Postman (<http://www.getpostman.com/>). Esta aplicación, integrada dentro del navegador Google Chrome, facilita el envío de peticiones HTTP con cualquier tipo de verbo y muestra la respuesta en varios formatos.

6.2.3. Framework PHP

La elección del framework PHP, entre la gran cantidad de los existentes actualmente en el mercado, ha sido la de Laravel 4.1 (<http://www.laravel.com>).

Este framework ha alcanzado una gran popularidad en los últimos meses ¹ debido a, entre otras características, la potencia que presta a la hora de crear APIs, la claridad del código que genera (tanto PHP como html para las vistas) y el hecho de ser una evolución de symfony, que ha sido uno de los dominadores en el campo de los frameworks PHP de los últimos años.

Para el caso particular de este proyecto, es muy importante la posibilidad de creación de APIs con Laravel, ya que esto significa que se podrá utilizar el mismo framework para la API y el front-end web, reduciendo el tiempo necesario en formación durante la etapa de desarrollo.

En Laravel también se pueden instalar librerías externas, y se ha hecho uso de una en concreto, denominada laravel-cors. CORS significa *Cross-Origin Resource Sharing*, y debe ser tenido en cuenta al desarrollar una API pública. La librería instalada configura las cabeceras de las respuestas de este servidor con los valores adecuados de CORS, de forma que pueda recibir peticiones XMLHttpRequest desde cualquier servidor. Sin esta configuración, BioVoiceWEB solo funcionaría si estuviera incluida en el mismo dominio que BioVoiceAPI.

6.2.4. Librerías Android

Al realizar una aplicación nativa para dispositivos Android, la tecnología de la que disponemos está ya cerrada. La programación se desarrollará con Java como lenguaje, utilizando el SDK de Google para programación en Android. Sí que podemos mencionar, no obstante, las librerías externas utilizadas para agilizar el tiempo de creación de esta aplicación:

- **Retrofit**: Facilita la comunicación mediante HTTP con APIs REST. Se encarga, además, de convertir los ficheros JSON a clases Java mediante el uso de GSON de Google (de forma transparente al programador). <http://square.github.io/retrofit/>
- **Android Saripaar**: Permite realizar validación de formularios en una actividad o fragmento de una app Android. Nos permite no tener que implementar desde cero una tarea tan repetitiva como esta, además de contar con un código fuente más limpio, ya que se basa en el uso de anotaciones, muy auto explicativas y poco intrusivas. <https://github.com/ragunathjawahar/android-saripaar>

¹A fecha de Junio de 2014

- **android-circlebutton**: Proporciona un tipo de botón circular con efectos al ser pulsada. No es en absoluto esencial, pero mejora la estética de la aplicación. <https://github.com/markushi/android-circlebutton>
- **HoloGraphLibrary**: Añade gráficas al conjunto de vistas disponible para Android. Se utiliza para mostrar de forma visual la comparación entre el resultado de una verificación de usuario y el valor umbral que decide su autenticidad o no. Una vez más, no es una librería crítica, pero en este caso es realmente importante, ya que muestra un concepto complejo de forma sencilla.

6.2.5. Documentación interna

Todo el código desarrollado está debidamente documentado en formato HTML, utilizando diferentes paquetes de generación de documentación automática. En el caso de las aplicaciones PHP se ha utilizado PHPDoc, mientras que para Java, se ha utilizado el estándar JavaDoc.

Se entrega la documentación generada, acompañando al código fuente. En el capítulo 1 se puede encontrar su ubicación exacta para cada componente.

6.3. Control de versiones y gestión de dependencias

El control de versiones se ha realizado enteramente en GIT. El uso de este tipo de herramientas no sólo ha proporcionado en este proyecto sus ventajas habituales de seguridad ante pérdidas de datos o versiones incorrectas, sino que se han utilizado repositorios accesibles via web, lo que facilita sobremanera el despliegue y la instalación de cada paquete desarrollado en nuevas máquinas.

Para la gestión de dependencias debemos separar, una vez más, los componentes realizados en PHP de los realizados en Java. Para la instalación de dependencias en PHP se ha utilizado Composer (<https://getcomposer.org/>). En el caso de java se ha utilizado Gradle (<http://www.gradle.org/>), que no sólo se encarga de incluir las dependencias de nuestro proyecto, sino también de establecer las opciones de configuración adecuadas para su compilación en forma de aplicación android.

6.4. Formación

Los componentes que más formación han requerido por parte del alumno han sido el framework Laravel y Alize, dado que en ninguno de los dos sistemas se tenía ningún tipo de experiencia previa.

En el caso de Laravel la formación se ha realizado mediante el uso de [Sau14] y [Ree14]. De estas dos referencias bibliográficas se realizó una lectura inicial y, posteriormente, se han utilizado como referencia durante todo el desarrollo del proyecto.

En el caso de Alize las fuentes son más variadas. Los propios autores han publicado dos papers ([LBF⁺13] y [BSM⁺08]), que presentan conceptos básicos pero no ahondan en cuestiones técnicas. Para ver en detalle los parámetros de configuración de cada ejecutable debemos acudir a la sección de documentación en la página web del proyecto (http://alize.univ-avignon.fr/doc_en.html).

También Composer y Gradle, los dos gestores de dependencias, han sido utilizados por primera vez, pero en estos casos su curva de aprendizaje, para las funciones básicas que se han utilizado, ha sido muy rápida.

Capítulo 7.

Pruebas

7.1. Introducción

La necesidad de realizar y documentar pruebas en el proyecto desarrollado es más que evidente, teniendo en cuenta la dimensión del mismo y la cantidad de partes involucradas. El objetivo de estas pruebas es garantizar el correcto funcionamiento, tanto de cada parte, como del sistema a nivel global, respecto a los requisitos encontrados en el análisis.

7.2. Plan de pruebas

Podemos distinguir dos niveles de pruebas necesarias. Por una parte, se deberán realizar pruebas unitarias de los componentes que las soporten, en este caso “alizabeth” y “BioVoiceAPI”, mientras que para los componentes restantes, consistentes en los front-ends que hacen uso de los anteriores, se realizarán pruebas de integración, que comprueben la correcta funcionalidad del sistema como un conjunto, involucrando a todas las partes que componen el proyecto.

Respecto a la técnica de prueba, las diferentes particularidades de cada componente hacen que sean necesarias, o bien pruebas de caja blanca, o bien de caja negra. Se especificará, en cada caso, la técnica elegida.

Finalmente, para desarrollar las pruebas unitarias, se utilizará PHPUnit en los componentes PHP y JUnit en la app Android. La referencia principal utilizada para realizar los desarrollos de los tests unitarios ha sido [Osh13].

7.3. Pruebas unitarias: alizephp

En esta sección se presentan las pruebas unitarias realizadas en el módulo “alizephp”. La técnica escogida ha sido la de pruebas de caja blanca, por lo que no dividiremos las entradas en casos de equivalencia.

Cada caso de prueba se corresponde con una prueba unitaria desarrollada en el código de alizephp. Tanto este documento como el código fuente se pueden relacionar mediante el identificador de cada caso de prueba (T-ALI x , donde x se corresponde con el número de caso de prueba), especificado en ambos ficheros.

7.3.1. Casos de prueba

T-ALI01 - Nombre de usuario vacío

Tabla 7.1.: T-ALI01 - Nombre de usuario vacío

Casos de uso afectados	CU02, CU03		
Precondicioness	El nombre de usuario es una cadena vacía		
Postcondiciones	Se genera un error		
Test pasado/fallado	✓		
Ejecuciones	Num.	Res.	Observaciones
	1	OK	-

T-ALI02 - Nombre de usuario compuesto por caracteres en blanco

Tabla 7.2.: T-ALI02 - Nombre de usuario compuesto por caracteres en blanco

Casos de uso afectados	CU02, CU03		
Precondicioness	El nombre de usuario tiene caracteres en blanco únicamente		
Postcondiciones	Se genera un error		
Test pasado/fallado	✓		
Ejecuciones	Num.	Res.	Observaciones
	1	Error	Añadido método trim()
	2	OK	-

T-ALI03 - Creación correcta de usuario

Tabla 7.3.: T-ALI03 - Creación correcta de usuario

Casos de uso afectados	CU02, CU03		
Precondicioness	El nombre de usuario es válido. El nombre de fichero pertenece a un fichero existente y con permisos de escritura		
Postcondiciones	Se crea una instancia en memoria de la clase AlizePHP, cuyo nombre de usuario y fichero de audio son los indicados		
Test pasado/fallado	✓		
Ejecuciones	Num.	Res.	Observaciones
	1	OK	-

T-ALI04 - Extracción de características con nombre de fichero vacío

Tabla 7.4.: T-ALI04 - Extracción de características con nombre de fichero vacío

Casos de uso afectados	CU02, CU03		
Precondicioness	El nombre de fichero es una cadena de texto vacía		
Postcondiciones	Se genera un error		
Test pasado/fallado	✓		
Ejecuciones	Num.	Res.	Observaciones
	1	OK	-

T-ALI05 - Extracción de características con fichero inexistente

Tabla 7.5.: T-ALI05 - Extracción de características con fichero inexistente

Casos de uso afectados	CU02, CU03		
Precondicioness	El nombre de fichero pertenece a un fichero inexistente en el sistema		
Postcondiciones	Se genera un error		
Test pasado/fallado	✓		
Ejecuciones	Num.	Res.	Observaciones
	1	OK	-

T-ALI06 - Extracción de características con fichero sin permisos de lectura

Tabla 7.6.: T-ALI06 - Extracción de características con fichero sin permisos de lectura

Casos de uso afectados	CU02, CU03		
Precondicioness	El nombre de fichero pertenece a un fichero sin permisos de lectura para el proceso que ejecuta el script		
Postcondiciones	Se genera un error		
Test pasado/fallado	✓		
Ejecuciones	Num.	Res.	Observaciones
	1	OK	-

T-ALI07 - Extracción de características correcta

Tabla 7.7.: T-ALI07 - Extracción de características correcta

Casos de uso afectados	CU02, CU03		
Precondicioness	El fichero de audio del usuario existe y el proceso que ejecuta el script cuenta con permisos de lectura sobre él		
Postcondiciones	Se crea un fichero de características		
Test pasado/fallado	✓		
Ejecuciones	Num.	Res.	Observaciones
	1	OK	-

T-ALI08 - Normalización de energía

Tabla 7.8.: T-ALI08 - Normalización de energía

Casos de uso afectados	CU02, CU03		
Precondicioness	El fichero de características del usuario existe y el proceso que ejecuta el script cuenta con permisos de lectura sobre él		
Postcondiciones	Se crea un fichero de características con la energía normalizada		
Test pasado/fallado	✓		
Ejecuciones	Num.	Res.	Observaciones
	1	OK	-

T-ALI09 - Normalización de energía sin fichero de características

Tabla 7.9.: T-ALI09 - Normalización de energía sin fichero de características

Casos de uso afectados	CU02, CU03		
Precondicioness	No existe el fichero de características del usuario o el proceso que ejecuta el script no cuenta con permisos de lectura sobre él		
Postcondiciones	Se genera una excepción		
Test pasado/fallado	✓		
Ejecuciones	Num.	Res.	Observaciones
	1	OK	-

T-ALI10 - Detección de energía

Tabla 7.10.: T-ALI10 - Detección de energía

Casos de uso afectados	CU02, CU03		
Precondicioness	El fichero de características con la energía normalizada del usuario existe y el proceso que ejecuta el script cuenta con permisos de lectura sobre él		
Postcondiciones	Se crea un fichero de etiquetas señalando las partes del audio con voz		
Test pasado/fallado	✓		
Ejecuciones	Num.	Res.	Observaciones
	1	OK	-

T-ALI11 - Detección de energía sin fichero de características normalizadas

Tabla 7.11.: T-ALI11 - Detección de energía sin fichero de características normalizadas

Casos de uso afectados	CU02, CU03		
Precondicioness	No existe el fichero de características con la energía normalizada del usuario o el proceso que ejecuta el script no cuenta con permisos de lectura sobre él		
Postcondiciones	Se genera una excepción		
Test pasado/fallado	✓		
Ejecuciones	Num.	Res.	Observaciones
	1	OK	-

T-ALI12 - Normalización de características

Tabla 7.12.: T-ALI12 - Normalización de características

Casos de uso afectados	CU02, CU03		
Precondicioness	El fichero de características con la energía normalizada del usuario y el de etiquetas existen y el proceso que ejecuta el script cuenta con permisos de lectura sobre ellos		
Postcondiciones	Se crea un fichero de características normalizadas		
Test pasado/fallado	✓		
Ejecuciones	Num.	Res.	Observaciones
	1	OK	-

T-ALI13 - Normalización de características sin ficheros necesarios

Tabla 7.13.: T-ALI13 - Normalización de características sin ficheros necesarios

Casos de uso afectados	CU02, CU03		
Precondicioness	No existe el fichero de características con la energía normalizada del usuario y/o el fichero de etiquetas, o el proceso que ejecuta el script no cuenta con permisos de lectura sobre alguno de ellos		
Postcondiciones	Se genera una excepción		
Test pasado/fallado	✓		
Ejecuciones	Num.	Res.	Observaciones
	1	OK	-

T-ALI14 - Creación del modelo

Tabla 7.14.: T-ALI14 - Creación del modelo

Casos de uso afectados	CU02		
Precondicioness	El fichero de características normalizadas existe y el proceso que ejecuta el script cuenta con permisos de lectura sobre él		
Postcondiciones	Se crea un fichero de modelo		
Test pasado/fallado	✓		
Ejecuciones	Num.	Res.	Observaciones
	1	OK	-

T-ALI15 - Creación de modelo sin ficheros necesarios

Tabla 7.15.: T-ALI15 - Creación de modelo sin ficheros necesarios

Casos de uso afectados	CU03		
Precondicioness	No existe el fichero de características normalizadas del usuario o el proceso que ejecuta el script no cuenta con permisos de lectura sobre él		
Postcondiciones	Se genera una excepción		
Test pasado/fallado	✓		
Ejecuciones	Num.	Res.	Observaciones
	1	OK	-

T-ALI16 - Verificación de usuario

Tabla 7.16.: T-ALI16 - Verificación de usuario

Casos de uso afectados	CU03		
Precondicioness	Existen el modelo del usuario del que se quiere comprobar las características y se han extraído las características normalizadas del audio que, supuestamente, pertenece a ese usuario		
Postcondiciones	Se crea un fichero de resultados		
Test pasado/fallado	✓		
Ejecuciones	Num.	Res.	Observaciones
	1	OK	-

T-ALI17 - Verificación de usuario sin modelo

Tabla 7.17.: T-ALI17 - Verificación de usuario sin modelo

Casos de uso afectados	CU03		
Precondicioness	No existe el modelo del usuario sobre el que se quiere realizar la verificación de locutor, o el proceso del script no tiene permisos de lectura sobre él		
Postcondiciones	Se genera una excepción		
Test pasado/fallado	✓		
Ejecuciones	Num.	Res.	Observaciones
	1	OK	-

T-ALI18 - Verificación de usuario sin fichero de características

Tabla 7.18.: T-ALI18 - Verificación de usuario sin fichero de características

Casos de uso afectados	CU03		
Precondicioness	No existe el fichero de características del audio sobre el que se quiere realizar la verificación, o el proceso del script no tiene permisos de lectura sobre él		
Postcondiciones	Se genera una excepción		
Test pasado/fallado	✓		
Ejecuciones	Num.	Res.	Observaciones
	1	OK	-

T-ALI19 - Borrado de usuario

Tabla 7.19.: T-ALI19 - Borrado de usuario

Casos de uso afectados	CU02, CU03		
Precondicioness	Ninguna		
Postcondiciones	Todos los ficheros relativos a un usuario que puedan existir en el sistema habrán sido borrados		
Test pasado/fallado	✓		
Ejecuciones	Num.	Res.	Observaciones
	1	Error	El fichero de modelo del usuario no se ha borrado
	2	OK	-

7.4. Pruebas unitarias y de integración: BioVoiceAPI

Las pruebas de la API se realizarán utilizando también tests unitarios. En este caso, el framework PHP utilizado, Laravel, añade una capa de abstracción a PHPUnit, permitiéndonos realizar llamadas HTTP de todo tipo y, posteriormente, obtener y analizar la respuesta recibida.

La técnica será la de pruebas de caja negra. Nos encargaremos de realizar tests que prueben todas las interfaces dispuestas por la API Rest al exterior, sin preocuparnos de la implementación. Acompañaremos, pues, a las pruebas, de sus particiones de equivalencia correspondientes.

En el caso de BioVoiceAPI, debemos tener claro que las pruebas de reconocimiento de locutor, al usar el sistema externo que proporciona AlizePHP, son pruebas de integración, y no pruebas unitarias. En el caso de las pruebas de registro de usuario, a pesar de utilizar la base de datos, las consideraremos pruebas unitarias¹.

7.4.1. Registro de usuario

En primer lugar, estudiaremos las particiones de equivalencia referidas a la gestión de usuarios, reflejadas en la tabla 7.20.

¹Este es un tema abierto a debate, dependiendo de la literatura consultada, las pruebas realizadas mediante el uso de bases de datos pueden llegar a ser consideradas pruebas unitarias o de integración

Tabla 7.20.: Particiones de equivalencia - Registro de usuario

Condición	Casos válidos	Casos no válidos
Nombre de usuario	Cadena de texto no registrada por otro usuario anteriormente, con mínimo 2 y máximo 6 caracteres y compuesta por letras, números, guiones y guiones bajos únicamente (1)	Cadena de texto vacía (5)
		Menos de 2 caracteres (6)
		Más de 6 caracteres (7)
		Nombre de usuario ya utilizado en un registro anterior (8)
		Cadena de texto con algún caracter distinto a letras, números, guiones o guiones bajos (9)
Nombre	Cadena de texto, formada únicamente por letras o espacios, de máximo 30 caracteres (2)	Cadena de texto vacía (10)
		Más de 30 caracteres (11)
		Algún caracter distinto de letra o espacio (12)
Apellidos	Cadena de texto, formada únicamente por letras o espacios, de máximo 100 caracteres (3)	Cadena de texto vacía (13)
		Más de 100 caracteres (14)
		Algún caracter distinto de letra o espacio (15)
Correo electrónico	Cadena de texto con correo electrónico válido, no registrada previamente para otro usuario y máximo 50 caracteres (4)	Cadena de texto vacía (16)
		Cadena de texto con correo electrónico no válido (17)
		Más de 50 caracteres (18)
		Correo electrónico previamente registrado para otro usuario (19)

Para, a continuación, realizar los casos de prueba necesarios. Una vez más, para cada caso de prueba ha sido desarrollado un test unitario en PHPUnit, estando ambos relacionados por el identificador T-API x .

T-API01 - Registro de usuario correcto

Tabla 7.21.: T-API01 - Registro de usuario correcto

Casos de uso afectados	CU01		
Particiones cubiertas	1, 2, 3, 4		
Precondicioness	Ninguna		
Postcondiciones	El usuario es creado en la base de datos		
Test pasado/fallado	✓		
Ejecuciones	Num.	Res.	Observaciones
	1	OK	-

T-API02 - Registro de usuario con nombre de usuario vacío

Tabla 7.22.: T-API02 - Registro de usuario con nombre de usuario vacío

Casos de uso afectados	CU01		
Particiones cubiertas	5		
Precondicioness	El nombre de usuario es una cadena vacía		
Postcondiciones	Error		
Test pasado/fallado	✓		
Ejecuciones	Num.	Res.	Observaciones
	1	OK	-

T-API03 - Registro de usuario con nombre de usuario menor de dos caracteres

Tabla 7.23.: T-API03 - Registro de usuario con nombre de usuario menor de dos caracteres

Casos de uso afectados	CU01		
Particiones cubiertas	6		
Precondicioness	El nombre de usuario tiene menos de dos caracteres		
Postcondiciones	Error		
Test pasado/fallado	✓		
Ejecuciones	Num.	Res.	Observaciones
	1	OK	-

T-API04 - Registro de usuario con nombre de usuario mayor de seis caracteres

Tabla 7.24.: T-API04 - Registro de usuario con nombre de usuario mayor de seis caracteres

Casos de uso afectados	CU01		
Particiones cubiertas	7		
Precondicioness	El nombre de usuario tiene más de seis caracteres		
Postcondiciones	Error		
Test pasado/fallado	✓		
Ejecuciones	Num.	Res.	Observaciones
	1	OK	-

T-API05 - Registro de usuario con nombre de usuario ya registrado

Tabla 7.25.: T-API05 - Registro de usuario con nombre de usuario ya registrado

Casos de uso afectados	CU01		
Particiones cubiertas	8		
Precondicioness	El nombre de usuario ya ha sido registrado en la base de datos		
Postcondiciones	Error		
Test pasado/fallado	✓		
Ejecuciones	Num.	Res.	Observaciones
	1	OK	-

T-API06 - Registro de usuario con nombre de usuario inválido

Tabla 7.26.: T-API06 - Registro de usuario con nombre de usuario inválido

Casos de uso afectados	CU01		
Particiones cubiertas	9		
Precondicioness	El nombre de usuario contiene caracteres distintos a letras, números, guiones o guiones bajos		
Postcondiciones	Error		
Test pasado/fallado	✓		
Ejecuciones	Num.	Res.	Observaciones
	1	OK	-

T-API07 - Registro de usuario con nombre vacío

Tabla 7.27.: T-API07 - Registro de usuario con nombre vacío

Casos de uso afectados	CU01		
Particiones cubiertas	10		
Precondicioness	El nombre debe ser una cadena en blanco, o formada solo por blancos		
Postcondiciones	Error		
Test pasado/fallado	✓		
Ejecuciones	Num.	Res.	Observaciones
	1	OK	-

T-API08 - Registro de usuario con nombre mayor de 30 caracteres

Tabla 7.28.: T-API08 - Registro de usuario con nombre mayor de 30 caracteres

Casos de uso afectados	CU01		
Particiones cubiertas	11		
Precondicioness	El nombre debe ser una cadena de más de 30 caracteres		
Postcondiciones	Error		
Test pasado/fallado	✓		
Ejecuciones	Num.	Res.	Observaciones
	1	Error	No se había especificado el tipo de entrada como cadena
	2	OK	-

T-API09 - Registro de usuario con nombre inválido

Tabla 7.29.: T-API09 - Registro de usuario con nombre inválido

Casos de uso afectados	CU01		
Particiones cubiertas	12		
Precondicioness	El nombre debe ser una cadena en blanco con algún caracter distinto de letra o espacio		
Postcondiciones	Error		
Test pasado/fallado	✓		
Ejecuciones	Num.	Res.	Observaciones
	1	OK	-

T-API10 - Registro de usuario con apellido vacío

Tabla 7.30.: T-API10 - Registro de usuario con apellido vacío

Casos de uso afectados	CU01		
Particiones cubiertas	13		
Precondicioness	El apellido debe ser una cadena en blanco, o formada solo por blancos		
Postcondiciones	Error		
Test pasado/fallado	✓		
Ejecuciones	Num.	Res.	Observaciones
	1	OK	-

T-API11 - Registro de usuario con apellido mayor de 100 caracteres

Tabla 7.31.: T-API11 - Registro de usuario con apellido mayor de 100 caracteres

Casos de uso afectados	CU01		
Particiones cubiertas	14		
Precondicioness	El apellido debe ser una cadena de más de 100 caracteres		
Postcondiciones	Error		
Test pasado/fallado	✓		
Ejecuciones	Num.	Res.	Observaciones
	1	Error	No se había especificado el tipo de entrada como cadena
	2	OK	-

T-API12 - Registro de usuario con apellido inválido

Tabla 7.32.: T-API12 - Registro de usuario con apellido inválido

Casos de uso afectados	CU01		
Particiones cubiertas	15		
Precondicioness	El apellido debe ser una cadena de más de 100 caracteres		
Postcondiciones	Error		
Test pasado/fallado	✓		
Ejecuciones	Num.	Res.	Observaciones
	1	OK	-

T-API13 - Registro de usuario con email vacío

Tabla 7.33.: T-API13 - Registro de usuario con email vacío

Casos de uso afectados	CU01		
Particiones cubiertas	16		
Precondicioness	El email debe ser una cadena de vacía o formada por solo blancos		
Postcondiciones	Error		
Test pasado/fallado	✓		
Ejecuciones	Num.	Res.	Observaciones
	1	OK	-

T-API14 - Registro de usuario con email inválido

Tabla 7.34.: T-API14 - Registro de usuario con email inválido

Casos de uso afectados	CU01		
Particiones cubiertas	17		
Precondicioness	El email no debe ser un email válido		
Postcondiciones	Error		
Test pasado/fallado	✓		
Ejecuciones	Num.	Res.	Observaciones
	1	OK	-

T-API15 - Registro de usuario con email mayor de 50 caracteres

Tabla 7.35.: T-API15 - Registro de usuario con email mayor de 50 caracteres

Casos de uso afectados	CU01		
Particiones cubiertas	17		
Precondicioness	El email debe ser una cadena de más de 50 caracteres		
Postcondiciones	Error		
Test pasado/fallado	✓		
Ejecuciones	Num.	Res.	Observaciones
	1	OK	-

T-API16 - Registro de usuario con email registrado previamente

Tabla 7.36.: T-API16 - Registro de usuario con email registrado previamente

Casos de uso afectados	CU01		
Particiones cubiertas	17		
Precondicioness	El email debe haber sido almacenado en la base de datos para algún otro usuario		
Postcondiciones	Error		
Test pasado/fallado	✓		
Ejecuciones	Num.	Res.	Observaciones
	1	OK	-

7.4.2. Inscripción de locutor

A continuación, serán estudiadas las particiones de equivalencia referidas al reconocimiento de locutor, reflejadas en la tabla 7.37, y los casos de prueba necesarios.

Tabla 7.37.: Particiones de equivalencia - Inscripción de locutor

Condición	Casos válidos	Casos no válidos
Inscripción de locutor	Usuario registrado y fichero de audio válido (1)	Usuario no registrado (3)
		Fichero de audio no válido (4)

T-API17 - Inscripción de locutor

Tabla 7.38.: T-API17 - Inscripción de locutor

Casos de uso afectados	CU02,CU03		
Particiones cubiertas	1		
Precondicioness	El usuario debe haber sido registrado en la base de datos previamente y el fichero de audio enviado es válido.		
Postcondiciones	Generado el modelo de voz del usuario		
Test pasado/fallado	✓		
Ejecuciones	Num.	Res.	Observaciones
	1	OK	-

T-API18 - Inscripción de locutor con usuario no registrado

Tabla 7.39.: T-API18 - Inscripción de locutor con usuario no registrado

Casos de uso afectados	CU02,CU03		
Particiones cubiertas	1		
Precondicioness	El usuario no debe haber sido registrado en la base de datos		
Postcondiciones	Error		
Test pasado/fallado	✓		
Ejecuciones	Num.	Res.	Observaciones
	1	Error	Se genera una excepción PHP, no una respuesta JSON con el error
	2	OK	-

T-API19 - Inscripción de locutor con fichero inválido

Tabla 7.40.: T-API19 - Inscripción de locutor con fichero inválido

Casos de uso afectados	CU02,CU03		
Particiones cubiertas	1		
Precondicioness	El fichero no debe ser válido, es decir, debe tener un formato de audio no soportado o no tener concedidos permisos de lectura al proceso que ejecuta el script.		
Postcondiciones	Error		
Test pasado/fallado	✓		
Ejecuciones	Num.	Res.	Observaciones
	1	Error	Se genera una excepción PHP, no una respuesta JSON con el error
	2	OK	-

7.4.3. Verificación de locutor

A continuación, serán estudiadas las particiones de equivalencia referidas a la verificación de locutor, reflejadas en la tabla 7.41.

Tabla 7.41.: Particiones de equivalencia - Verificación de locutor

Condición	Casos válidos	Casos no válidos
Verificación de locutor	Usuario a verificar inscrito y fichero de audio válido (2)	Usuario a verificar no inscrito (5)
		Fichero de audio no válido (6)

T-API20 - Verificación de locutor

Tabla 7.42.: T-API20 - Verificación de locutor

Casos de uso afectados	CU02,CU03		
Particiones cubiertas	1		
Precondicioness	El usuario debe estar inscrito en el sistema y el fichero de audio enviado es válido.		
Postcondiciones	Calculado valor de verificación de locutor		
Test pasado/fallado	✓		
Ejecuciones	Num.	Res.	Observaciones
	1	OK	-

T-API21 - Verificación de locutor con usuario no inscrito

Tabla 7.43.: T-API21 - Verificación de locutor con usuario no inscrito

Casos de uso afectados	CU02,CU03		
Particiones cubiertas	1		
Precondicioness	El usuario no debe haber sido inscrito en el sistema		
Postcondiciones	Error		
Test pasado/fallado	✓		
Ejecuciones	Num.	Res.	Observaciones
	1	Error	Se genera una excepción PHP, no una respuesta JSON con el error
	2	OK	-

T-API22 - Verificación de locutor con fichero inválido

Tabla 7.44.: T-API22 - Verificación de locutor con fichero inválido

Casos de uso afectados	CU02,CU03		
Particiones cubiertas	1		
Precondicioness	El fichero no debe ser válido, es decir, debe tener un formato de audio no soportado o no tener concedidos permisos de lectura al proceso que ejecuta el script.		
Postcondiciones	Error		
Test pasado/fallado	✓		
Ejecuciones	Num.	Res.	Observaciones
	1	Error	Se genera una excepción PHP, no una respuesta JSON con el error
	2	OK	-

7.5. Pruebas de integración: BioVoiceWeb y BioVoiceAPP

Una vez realizados los tests unitarios y de integración en las capas inferiores de nuestra arquitectura, como son la API y alizephp, debemos testear el correcto funcionamiento de los front-ends que hemos desarrollado.

Cuestiones como la comprobación del correcto funcionamiento de la creación de usuarios o de la inscripción y verificación de locutor serían redundantes a este nivel, sólo debemos comprobar que los front-ends alcanzan la API, reciben las respuestas, y muestran los elementos pertinentes en consecuencia.

No tiene mucho sentido, por tanto, desarrollar tests unitarios para probar estos aspectos en los documentos HTML generados o los fragmentos de estas aplicaciones, por lo que la técnica que se utilizará es la de la comprobación del flujo de la aplicación, comprobando que, desde cada nodo se accede a los siguientes, dependiendo de las entradas y salidas recibidas.

El flujo de ambos front-ends es el mismo, por lo que no habrá diferencias en cuanto al diagrama que lo representa, reflejado en la figura 7.1.

La tabla 7.45 refleja los casos de prueba realizados para comprobar este flujo en ambas plataformas.

Figura 7.1.: Diagrama de flujo de los front-ends

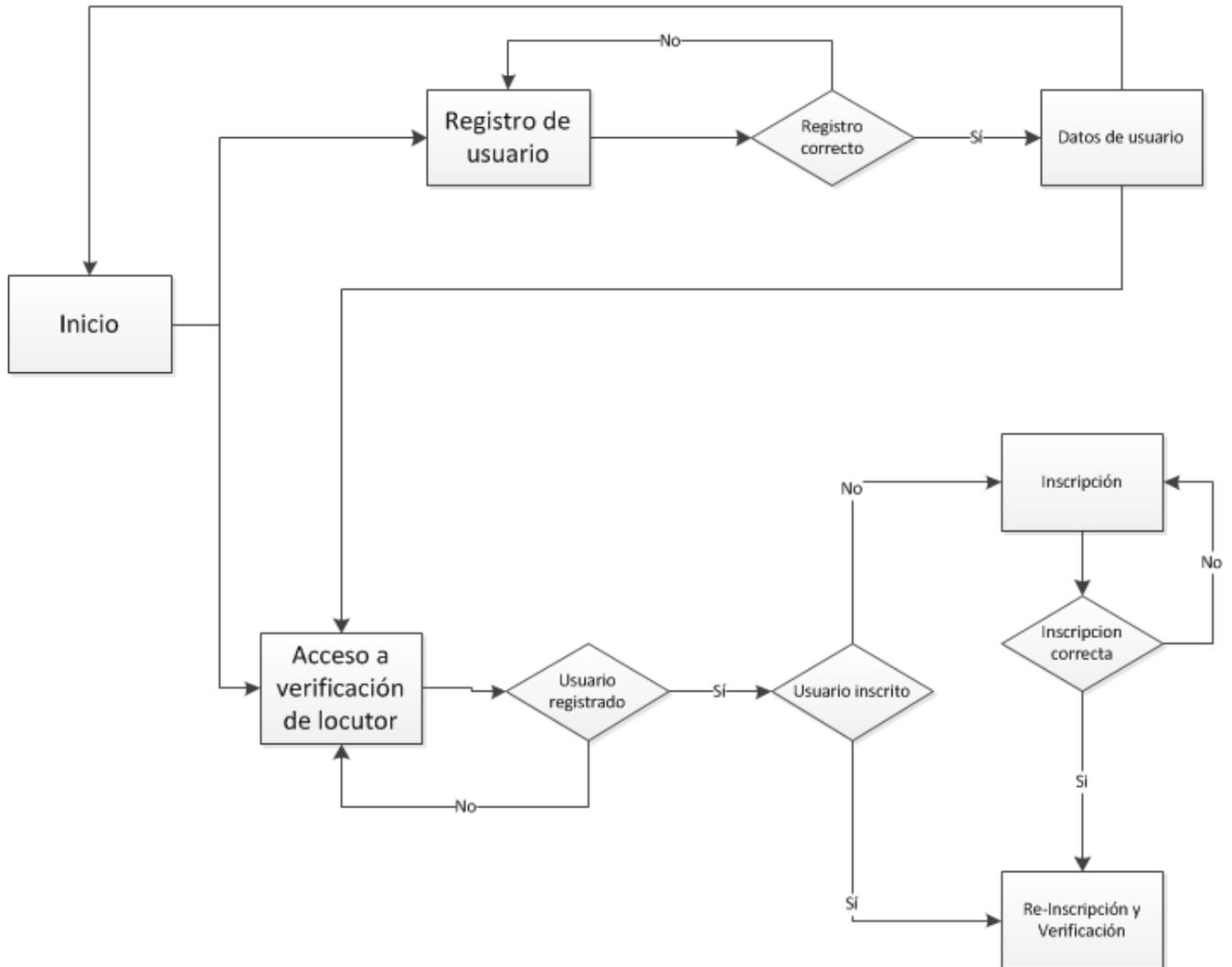


Tabla 7.45.: Casos de prueba - Flujo en front-ends

Estado	Postcondiciones	Estados destino	Web	App
Inicio (1)	Ninguna	2, 4	✓	✓
Registro de usuario (2)	Registro correcto	3	✓	✓
	Registro incorrecto	2	✓	✓
Datos de usuario (3)	Ninguna	4, 1	✓	✓
Acceso a verificación de locutor (4)	Usuario registrado no inscrito	5	✓	✓
	Usuario registrado e inscrito	6	✓	✓
	Usuario no registrado	4	✓	✓
Inscripción (5)	Inscripción correcta	6	✓	✓
	Inscripción no correcta	5	✓	✓
Re-Inscripción y Verificación (6)	Ninguna	6	✓	✓

Capítulo 8.

Control, seguimiento y plan de cierre

8.1. Introducción

El objetivo de esta sección es hacer una evaluación y valoración final sobre el proceso de elaboración del proyecto y los objetivos conseguidos. En este caso no se sigue estrictamente una plantilla, ya que al tratarse de un proyecto académico no es posible. Se omiten, por la misma razón, cuestiones presupuestarias y de gestión de recursos.

8.2. Objetivos conseguidos

En el momento de finalización de un proyecto es una práctica aconsejable realizar una valoración de los objetivos conseguidos, en relación a los que se pretendía conseguir. Esta es una buena medida del nivel de éxito en la realización del proyecto, aunque no la única, y menos en un proyecto académico. Se presentan, a continuación, una lista de los objetivos que se pretendían cumplir inicialmente y un análisis sobre si ha llegado o no a cumplirse.

- El principal objetivo de todo proyecto de ingeniería de software es el desarrollo del sistema en cuestión. Este objetivo se ha conseguido, disponemos de una web y una aplicación móvil que dan soporte a un sistema de reconocimiento de locutor.
- La actualización del sistema anterior exigía, mas allá de mantener la funcionalidad, que se utilizaran técnicas de programación modernas. Este objetivo se ha conseguido con creces. La versión 5 de PHP está preparada para desarrollar webs utilizando el paradigma de la orientación a objetos e incluye control de excepciones, por lo que las partes web del desarrollo cuentan con estas características. El uso, además, de Laravel como framework, ha facilitado la total separación de las vistas y la lógica de las aplicaciones, siguiendo el patrón modelo vista controlador. En el caso de la aplicación Android, el hecho de estar desarrollada en Java ya nos obliga a utilizar orientación a objetos y excepciones, así como el SDK de Android hace necesaria la separación de las vistas y la lógica de negocio.
- Un proyecto académico como este, que puede llegar a ser utilizado en el futuro por el grupo de investigación por el que ha sido encargado, o como base para otros trabajos de fin de grado, es importante que sea fácilmente modificable y escalable. La arquitectura diseñada permite esto, haciendo que sea muy sencillo crear nuevos front-ends o la adición de nuevos sistemas de reconocimiento de locutor.
- Finalmente, y relacionado con el objetivo anterior, es muy importante contar con una documentación de calidad, que permita a otros usuarios instalar, usar, y valorar el sistema. Esta memoria es parte de esa documentación, y junto con los manuales de usuario e instalación que le acompañan forma una completa documentación externa. A la documentación interna, por otra parte, referida a los comentarios en el código fuente, también se le ha prestado especial atención, generando para todos los componentes su versión en HTML.

8.3. Control y seguimiento del proyecto

Nos encontramos en esta sección toda la metodología y procesos seguidos que guardan relación con el seguimiento del proyecto con el fin de que este finalice de manera exitosa. Además se incluyen secciones que reflejan los problemas que se han tenido junto con un análisis de cómo se han solucionado y como han afectado al desarrollo.

8.3.1. Control de la programación temporal del proyecto

La programación temporal previa que se había realizado para llevar a cabo este proyecto ha sido imposible de conseguir. Las diferentes circunstancias ocurridas han hecho que la fecha de entrega de este proyecto se retrase de Julio a Septiembre.

Se analiza cada fase a continuación, si entrar en detalles relativos a los riesgos que han afectado al proyecto, que se detallan en la sección 8.3.4.

- **Plan de desarrollo:** El plan de desarrollo se realizó en un periodo de tiempo corto.
- **Estado del arte:** La estimación de la duración de esta fase ha resultado ser mucho menor de lo que debiera. El software utilizado, Alize, incluye dos sistemas de reconocimiento de locutor diferentes: uno basado en I-Vectors y otro basado en UBM/GMM. En principio, la primera alternativa es la que actualmente muestra un mejor rendimiento. Además, con el conjunto de datos de prueba que incluye Alize los resultados parecían ser buenos. Por lo tanto, fue la primera opción escogida. Se implementó nuestro sistema basado en ella y al realizar las pruebas de reconocimiento observamos una tasa de errores extrañamente alta. Probado el sistema mediante corpus biométricos de voz distintos al que incluye Alize, tanto por parte del tutor como por mi parte, se corroboró el mal funcionamiento de la parte de I-Vectors incluida en Alize. En este punto, se decidió cambiar de enfoque y probar con el sistema basado en UBM/GMM, que mostró muy buenos resultados, teniendo que cambiar toda la implementación a este nuevo sistema. Este imprevisto ha tenido una gran repercusión temporal, lo que retrasó en gran medida el avance del resto de partes.
- **Análisis:** La fase de análisis se realizó en menos tiempo de lo previsto, ya que al revisar el sistema a actualizar se descubrió que el número de requisitos de usuario era escaso.
- **Diseño:** El diseño ha llevado más horas de las previstas, ya que se han hecho varias iteraciones para llegar a la solución final.

- **Implementación:** Al tener un diseño final más complejo de lo que se esperaba, la implementación también ha llevado más tiempo del planeado. El hecho de tener que crear tecnologías en las que no se tenía experiencia, como la API Rest, ha influido también en este aumento de duración.
- **Pruebas:** La duración de las pruebas, incluido el desarrollo de los tests unitarios, fue estimada correctamente.
- **Documentación final:** El uso de diferentes plantillas ha permitido realizar una reducción del tiempo empleado en realizar la documentación final.

8.3.2. Artefactos obtenidos

Los hitos marcados en la planificación inicial suponían la entrega de un artefacto definido en el plan de desarrollo. Estas “entregas” han sido otra manera de supervisar si el proyecto transcurría correctamente. Recordamos todos estos artefactos:

- Plan de desarrollo
- Fichero MS Project
- Especificación de estado del arte
- Especificación de análisis
- Especificación de diseño
- Diagrama entidad-relación de la base de datos
- Script SQL de creación de la base de datos
- Componentes software
- Especificación de pruebas
- Manual de instalación y usuario

8.3.3. Reuniones

Se han realizado varias reuniones entre el alumno y el tutor del proyecto, con el fin de ir tratando distintos aspectos y abordar los diferentes problemas que se han ido presentando.

Como se puede ver a continuación, se ha ido llevando a cabo un control de los temas que se trataban en cada una de ellas.

- **Miércoles 18/12/2013:** Inicio del proyecto, recopilación de documentación y código fuente de la aplicación a actualizar.
- **Jueves 20/03/2014:** Definición de los límites del proyecto, revisión del análisis.
- **Miércoles 21/05/2014:** Revisión de los sistemas de reconocimiento de locutor estudiados y del estado del arte.
- **Viernes 06/06/2014:** Revisión del problemas con rendimiento de Alize. Se replantea la entrega para Septiembre en lugar de Julio.
- **Jueves 17/07/2014:** Estudio de la solución final en problemas con Alize.
- **Miércoles 30/07/2014:** Gestión de la documentación para la solicitud de defensa del trabajo de fin de grado.

Además de estas reuniones, la comunicación por email ha sido fluida, y ha servido para tratar temas que no era necesario tratar en persona.

8.3.4. Problemas encontrados

Los problemas que se han encontrado, y a los cuales nos hemos tenido que enfrentar a lo largo del desarrollo del proyecto, han sido especialmente la aparición de alguno de los riesgos identificados en el plan de desarrollo. Estos riesgos son los que se listan a continuación:

- **Riesgo 02 - Reducción de la disponibilidad de algún miembro del grupo (tabla 3.11):** La probabilidad de este riesgo estaba definida como alta, por lo que se daba por hecho que su ocurrencia era probable. Se dio el caso, por tanto, de que el alumno encargado de la realización del proyecto comenzó a trabajar y su disponibilidad para realizar el proyecto se vio seriamente mermada. Esta ha la principal razón de que la fecha de entrega del trabajo, en lugar de ser en Julio, haya sido en Septiembre.

- **Riesgo 06 - Problemas con el uso de las herramientas (tabla 3.15):** A la hora de probar el software de reconocimiento de locutor Alize, se constató que los resultados que se obtenían eran muy deficientes, prácticamente aleatorios. Aunque el objetivo del proyecto no es la correcta configuración del sistema de reconocimiento de locutor, sino la plataforma que da soporte a su uso, los resultados obtenidos hacían el sistema inservible. Se siguió el plan de acción asociado, en el que la ayuda del tutor como voz autorizada en el dominio fué de gran relevancia para probar exhaustivamente el software de reconocimiento de locutor y reorientar la forma de conseguir el correcto funcionamiento del proceso, cambiando la técnica utilizada de i-vectores a GMM/UBM.

Capítulo 9.

Conclusiones y trabajo futuro

9.1. Introducción

En este capítulo se desarrollan las conclusiones sacadas del proyecto realizado, las lecciones aprendidas y las líneas a seguir en un trabajo futuro.

9.2. Conclusiones

La funcionalidad con la que contaba el sistema a actualizar se ha implementado completamente, por lo que la finalización del proyecto se puede considerar totalmente exitosa.

No obstante, la funcionalidad no era el único aspecto a tener en cuenta a la hora de realizar este proyecto, sino que la actualización del sistema anterior se debía hacer siguiendo una línea de calidad en el desarrollo de software acorde al estado actual de esta disciplina. Este aspecto también se ha cuidado de forma especial, utilizando tecnologías modernas, que permiten la creación de un código limpio y bien estructurado. Se han seguido también una serie de buenas prácticas y convenciones, estándar de facto para cada uno de los lenguajes, que aumentan la calidad del proyecto final en general.

El diseño de la arquitectura, que ha supuesto un gran reto a la hora de desarrollar el producto software, hace que el proyecto vaya más allá de lo que se proponía inicialmente, ya que no sólo se implementa la solución con tecnologías de programación modernas, sino que se hace teniendo en cuenta criterios de calidad de ingeniería de software que permitirán, si es necesario, el futuro uso y modificación de este proyecto por parte de distintos implicados con relativa facilidad. El bajo acoplamiento de cada uno de los componentes y el uso de diferentes patrones software para permitir gran facilidad de mantenimiento y escalabilidad del sistema dan una idea del cuidado puesto en este apartado.

Por otro lado, la creación de pruebas unitarias, a pesar de aumentar la dificultad y el tiempo de desarrollo del proyecto, hace que se tenga una medida empírica de la solidez de los componentes más internos del sistema, como son el wrapper de alize para PHP y la API que da soporte al resto de la plataforma. El autor de este proyecto no había utilizado nunca pruebas unitarias para realizar los tests de un desarrollo, por lo que se ha descubierto su gran utilidad, no sólo a la hora de constatar el correcto funcionamiento del sistema, sino también como herramienta que ayuda a comprender mejor el proceso de prueba y completarlo sin dejar detalles sin observar.

Un aspecto clave a la hora de agilizar el proceso de desarrollo, ha sido la inclusión, de todos y cada uno de los componentes, en un repositorio software con un sistema de control de versiones. Esto permite tener una copia de seguridad siempre disponible de los proyectos, y poder comparar los cambios que se han ido realizando en ellos. Además, estos repositorios son usados por las herramientas de gestión de dependencias para descargar las mismas, haciendo que el proceso de instalación del proyecto se haya podido simplificar al máximo.

Por último, cabe destacar que, debido al escaso tiempo que ha tenido el autor de este proyecto, ha resultado imposible seguir la planificación inicial, así como las diferentes fases propuestas en el proceso unificado. Son muy valiosas las lecciones de necesidad de organización, cuidado a los detalles y documentación adquiridas, para poder llevar a cabo un proyecto de esta dimensión

en espacios de tiempo reducidos y separados entre si, así como la inestimable ayuda de una voz autorizada, en este caso el tutor del proyecto, para aclarar los aspectos más complejos, ahorrando una cantidad de tiempo considerable al alumno en su aprendizaje y comprensión.

Se puede concluir, por tanto, que el proyecto ha sido un éxito a nivel de aprendizaje, ya que se han manejado tanto una gran cantidad de nuevos conceptos técnicos para el alumno (tests unitarios, creación de APIs, gestión de dependencias, ...), como teóricos, en lo relativo a los sistemas de reconocimiento biométrico por voz.

9.3. Trabajo futuro

A pesar de que el producto final cumple todas las características deseables que se habían definido para este trabajo, hay una serie de mejoras que, de haber sido posible, se habrían incluido en el desarrollo final. No hay que olvidar que la asignación de tiempo para su realización es de 12 créditos (300 horas), por lo que aspectos que aporten poco a la formación del alumno, o que sean parte del dominio de otro problema, no pueden estar incluidos en el trabajo final. Algunos de ellos se presentan a continuación:

9.3.1. Identificación de usuarios

En el proyecto desarrollado, la API es totalmente pública, sin zonas privadas, por lo que un usuario no puede acceder a un área personal en la que tenga la opción de realizar las típicas acciones CRUD (Create, Read, Update, Delete) sobre sus propios datos.

Se planteó el desarrollo de estos componentes, y de hecho la API da soporte a la actualización, borrado y lectura de los datos de un usuario (aparte de la creación, que sí que forma parte del proyecto final), pero no estaba dentro de los requisitos originales del proyecto.

En todo caso, esta funcionalidad se podría implementar, en un futuro, o bien mediante un sistema de usuario y contraseña, o utilizando algún servidor de autenticación externo como los que proporciona OAuth. Si se planteara la creación del primer caso, sería más que aconsejable también la implementación de un servidor de certificados, para poder cifrar las conexiones HTTP a la API.

9.3.2. Mejora de resultados de verificación de locutor

Este aspecto pertenecería exclusivamente a la disciplina de la biométrica, y podría ser un trabajo a llevar a cabo a la hora de hacer un master o un doctorado. Se escapa, por tanto, del dominio en el que nos movemos en este proyecto. Aunque se ha intentado crear un sistema cuyos resultados, a la hora de realizar la verificación de locutor, sean los más fiables posible, hay ciertos aspectos que sólo han permitido alcanzar esta fiabilidad de manera parcial.

Por un lado, está la gran complejidad a la hora de analizar las muestras que presenta el hecho de que estén recogidas con diferentes micrófonos y en diferentes ambientes. Para poder implementar este sistema de manera correcta deberíamos tener varios modelos de fondo, con distintas características (micrófono de teléfono, de ordenador, ambiente de estudio, con ruido de fondo, ...) para

analizar cada muestra en el contexto adecuado, y esto es virtualmente imposible con los recursos de los que disponemos.

También los valores de configuración (cantidad de cepstrales, de gaussianas en el modelo o valores para la extracción de características) deberían ser replanteados, ya que los utilizados se basan en los experimentos realizados en un corpus de muestras concreto, que no tiene por qué corresponder con el real.

Bibliografía

- [BBF⁺04] Frédéric Bimbot, Jean-François Bonastre, Corinne Fredouille, Guillaume Gravier, Ivan Magrin-Chagnolleau, Sylvain Meignier, Teva Merlin, Javier Ortega-García, Dijana Petrovska-Delacrétaz, and Douglas A. Reynolds. A tutorial on text-independent speaker verification. *EURASIP J. Appl. Signal Process.*, 2004:430–451, January 2004.
- [BMM⁺04] Raphaël Blouet, Chafic Mokbel, Houda Mokbel, Eduardo Sánchez-Soto, Gérard Chollet, and Hanna Greige. Becars: a free software for speaker verification. In *Odyssey*, pages 145–148, 2004.
- [BSM⁺08] Jean-François Bonastre, Nicolas Scheffer, Driss Matrouf, Corinne Fredouille, Anthony Larcher, Alexandre Preti, Gilles Pouchoulin, Nicholas W. D. Evans, Benoit G. B. Fauve, and John S. D. Mason. Alize/spkdet: a state-of-the-art open source software for speaker recognition. In *Odyssey* [DBL08], page 20.
- [DBL08] *Odyssey 2008: The Speaker and Language Recognition Workshop, Stellenbosch, South Africa, January 21-24, 2008*. ISCA, 2008.
- [dIEyE98] Instituto de Ingeniería Eléctrica y Electrónica. 1058-1998 - IEEE Standard for Software Project Management Plans. Technical report, IEEE, 1998.
- [GHJV95] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-oriented Software*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995.
- [JPDCD09] A.K. Jain, D. Petrovska-Delacrétaz, G. Chollet, and B. Dorizzi. *Guide to Biometric Reference Systems and Performance Evaluation*. Springer, 2009.
- [Jr.97] Joseph P. Campbell Jr. Speaker recognition: A tutorial, 1997. <http://www.informatik.uni-ulm.de/ni/Lehre/SS06/PraktikumNI/Campbell.pdf>.
- [KESM14] E. Khoury, L. El Shafey, and S. Marcel. Spear: An open source toolbox for speaker recognition based on Bob. In *IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2014.

- [LBF⁺13] Anthony Larcher, Jean-François Bonastre, Benoit G. B. Fauve, Kong-Aik Lee, Christophe Lévy, Haizhou Li, John S. D. Mason, and Jean-Yves Parfait. Alize 3.0 - open source toolkit for state-of-the-art speaker recognition. In Frédéric Bimbot, Christophe Cerisara, Cécile Fougeron, Guillaume Gravier, Lori Lamel, François Pellegrino, and Pascal Perrier, editors, *INTERSPEECH*, pages 2768–2772. ISCA, 2013.
- [LJ09] S.Z. Li and A.K. Jain. *Encyclopedia of Biometrics: I - Z*. Number v. 1 in Encyclopedia of Biometrics. Springer, 2009.
- [Mon14] Polytechnique Montreal. *Unified Process for EDUcation*, 2014 (accessed August 5, 2014). <http://www.upedu.org/>.
- [Osh13] Roy Oshero. *The Art of Unit Testing, Second Edition: with examples in C#*. Manning Publications Co., Greenwich, CT, USA, 2nd edition, 2013.
- [RAR13] L. Richardson, M. Amundsen, and S. Ruby. *RESTful Web APIs*. O’Reilly Media, 2013.
- [Ree14] Dayle Rees. *Laravel: Code Bright*. Leanpub, 2014.
- [Sau14] R. Saunier. *Getting Started with Laravel 4*. Community experience distilled. Packt Publishing, 2014.

Apéndice A.

Contenido del CD

En el cd que incluye esta memoria se pueden encontrar el resto de componentes del proyecto. Se describe a continuación la estructura de carpetas que contiene, partiendo de la carpeta raíz, y qué se puede encontrar en cada una de ellas.

- **memoria.pdf:** El archivo que contiene este documento.
- **planificación.mpp:** Fichero de Microsoft Project 2013 con la definición, secuenciación y temporización de tareas, y los diagramas asociados.
- **src:** Carpeta con el código fuente del proyecto.
 - **alizephp:** Wrapper para Alize, software de reconocimiento biométrico.
 - **BioVoiceAPI:** Componente PHP encargado de recibir las peticiones HTTP de los front-ends y redirigirlas a los componentes apropiados.
 - **BioVoiceWeb:** Página web PHP que actúa como front-end de BioVoiceAPI.
 - **BioVoiceAPP:** Aplicación Android que actúa como front-end de BioVoiceAPI.
 - **config:** Ficheros de configuración relevantes.
- **bin:** Carpeta que contiene los binarios del proyecto. En el caso que nos ocupa, el único componente susceptible de tener versión compilada es BioVoiceAPP.
- **manuales:** Carpeta con los manuales de los diferentes componentes.
 - **instalación:** Manual de instalación de todos los componentes.
 - **usuario:** Manual de usuario de los dos front-ends desarrollados.
- **doc:** Carpeta con la documentación interna generada a partir del código fuente del proyecto. Se divide en subcarpetas con el nombre de cada componente relevante.

- **bdd:** Carpeta con los ficheros de base de datos necesarios.
 - **ER.mwb:** Diagrama entidad relación para MySQL WorkBench.
 - **bdd.sql:** Scrit SQL para crear la base de datos.

Apéndice B.

Versiones de este documento

Se listan las diferentes modificaciones que se han realizado sobre este documento (tabla B.1).

Tabla B.1.: Cambios en el documento

Versión	Fecha	Modificaciones
1.0	10/08/2014	Versión Inicial
1.1	23/08/2014	Añadidas pruebas de integración
1.2	28/08/2014	Primera revisión externa. Cambiada palabra cuadros por tablas, correcciones ortográficas.
1.5	02/09/2014	Revisión por parte del tutor. Corregidos objetivos generales y traducciones de términos matemáticos de inglés a español. Mejoras en la redacción de todos los apartados, correcciones ortográfica.