UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA

MÁSTER UNIVERSITARIO EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

Mecanismos de Interacción Enriquecidos con Técnicas de Computación Afectiva

José María García García

Febrero, 2019

UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA

Departamento de Sistemas Informáticos

TRABAJO FIN DE MÁSTER

Mecanismos de Interacción Enriquecidos con Técnicas de Computación Afectiva

Autor: José María García García

Director: Víctor Manuel Ruiz Penichet

Director: María Dolores Lozano Pérez

Resumen

Se entiende por Computación Afectiva toda aquella computación relacionada con, provocada por, o que influye en las emociones. En otras palabras, es toda aquella computación que tiene alguna relación con las emociones, ya sea porque se detectan o porque se simulan para algún fin, teniendo como objetivo, en este trabajo, la mejora de la experiencia interactiva de los usuarios. Ya en los inicios de esta disciplina, allá por 1995, la detección automática de emociones causó furor y fue una línea de investigación adoptada por muchos investigadores.

No obstante, en muy poco tiempo, todos esos investigadores que estaban trabajando en el campo de la detección llegaron a la misma conclusión: detectar emociones atendiendo solo a información de un tipo (información extraída de la expresión facial, de la voz, de señales fisiológicas producidas por el cuerpo, etc.) era un enfoque muy pobre. Nosotros, los humanos, somos los mejores detectores de emociones que existen. Esto se debe a que nosotros no utilizamos solo información de un canal afectivo: al interactuar con otra persona, no nos limitamos a analizar solo su voz o su cara, sino que la consideramos de forma global.

En un intento de imitar la forma de detectar emociones de los humanos, surgió la detección multimodal, esto es, la detección de emociones en base a varios canales de información. Tras las primeras pruebas de este tipo de detección, los investigadores de ese campo volvieron a coincidir en su veredicto: la detección multimodal era mejor, más precisa, que la detección unimodal, pero, al mismo tiempo, reconocían la dificultad y problemática que esta primera podía suponer (demasiadas tecnologías distintas, complejidad lógica, falta de estandarización, etc.).

A día de hoy, ese veredicto se mantiene, con sus correspondientes consecuencias: actualmente, la multimodalidad sigue relegada al apartado de «Trabajo futuro» de muchas investigaciones. Fruto de la revisión sistemática desarrollada en el marco de este Trabajo de Fin de Máster, se ha identificado que, si bien la gran mayoría de trabajos actuales están centrados en mejorar la precisión de la detección en un solo canal, los trabajos que se centran en realizar una aplicación real de la Computación Afectiva si utilizan la detección multimodal.

Este trabajo aborda algunos de los pilares sobre los que se sustenta esa reticencia a integrar la detección multimodal en trabajos actuales, ofreciendo un marco de trabajo (o *framework*) que facilite la integración de tecnologías de detección y el despliegue de sistemas multimodales completamente adaptados a las necesidades de sus usuarios. Con esto perseguimos que la aplicación real de la Computación Afectiva deje de representar una porción tan pequeña del esfuerzo total que se está invirtiendo en esta disciplina, puesto que el desarrollo de aplicaciones afectivas es el siguiente paso en el campo de la Interacción Persona-Ordenador.

Agradecimientos

Este trabajo constituye el umbral de una puerta que me decidí a cruzar hace años, la puerta al mundo de la docencia. Quisiera dar las gracias a todos los que, de alguna manera, me han ayudado a llegar hasta aquí, ya fuera ayudándome a apagar fuegos o simplemente escuchándome cuando necesitaba hablar con alguien.

En especial, quisiera dar las gracias a mis tutores, Víctor y María, que por mucho que confíen en mí, más confío yo en ellos, por su apoyo, su guía y su visión.

Dedicatorias

Quiero dedicar este proyecto a mis hermanas, Alicia y Lidia. Si algo he podido aprender de mis hermanas es que las emociones son algo complejo y totalmente multimodal.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1. INTRODUCCIÓN

En el siguiente capítulo se presentan las distintas motivaciones que han llevado a la realización de este trabajo, así como los objetivos que se persiguen durante el desarrollo del mismo. Se recoge también una explicación de la estructura y organización del presente documento.

1.1 Motivación

Se entiende por Computación Afectiva toda aquella computación relacionada con, provocada por, o que influye en emociones [1]. En otras palabras, es toda aquella computación que tiene alguna relación con las emociones, ya sea porque se detectan o porque se simulan. Esta línea de trabajo es un buen ejemplo de campo multidisciplinar, pues abarca la informática, la medicina y la psicología y además se ha aplicado en educación, sanidad, publicidad, entre otros campos. Si bien esta corriente surgió en 1995, actualmente nos encontramos en el periodo de tiempo en el cual se comienza a adoptar e integrar en nuestras vidas [2]. La actividad más identificativa de la Computación Afectiva es la *detección de emociones*: basándonos en las manifestaciones físicas de una emoción, estimamos qué está sintiendo una persona, mapeando los valores físicos medidos a una escala concreta, como puede ser un conjunto de categorías (alegría, tristeza, sorpresa, miedo, etc.) o una serie de dimensiones (valencia, excitación, control, etc.).

El principal problema que existe en la detección de emociones es que aún no podemos traducir la actividad del *sistema límbico*, la estructura del cerebro causante de las emociones, a una *medida precisa*, sino que tenemos que medir las emociones a través de su manifestación en la persona: expresión facial, tono y timbre de la voz, gestos de las manos, postura, diámetro de las pupilas, ritmo de la respiración, sudoración en la piel, etc. Todas estas medidas tienen *defectos*, y no siempre sus resultados son precisos. A veces se debe a la naturaleza del propio canal de información, que puede ser difícil de medir o interpretar; otras, a la propia persona, que puede ocultar sus emociones conscientemente. Para mejorar esa tasa de acierto todo lo que sea posible, se recurre a los sistemas multimodales [3]. Un **sistema multimodal** es aquel que combina los resultados de distintos detectores de emociones para obtener una medida más fiable. Si un sistema multimodal detecta la misma emoción a través de dos canales distintos, reafirma dicha detección, lo que supone en última instancia una mayor tasa de acierto, puesto que los detectores de emociones son, en esencia, *clasificadores automáticos*.

La propuesta de este Trabajo de Fin de Máster (en adelante, TFM) consiste en crear **un sistema multimodal a dos niveles**. Dado un conjunto de canales afectivos previamente seleccionados (cara, voz, postura, señales fisiológicas, etc.), crearemos un *sistema de validación monocanal* para cada uno de esos canales, lo que constituye el *primer nivel* del sistema. Se denomina **sistema de validación monocanal** a un sistema que toma varias medidas de un mismo canal usando tecnologías y/o servicios distintos. De esta manera, no solo se aumenta la precisión de la detección en ese canal (mediante la fusión de resultados) sino que también se realiza una validación cruzada entre los mismos, comparando todos los resultados entre sí para detectar posibles inexactitudes. En el *segundo nivel* del sistema se combinan los resultados combinados de cada canal para, una vez más, aumentar la confianza de cada clasificación y poder *detectar incongruencias o emociones complejas* que no se pueden detectar analizando solo un canal. Por ejemplo, si la expresión facial

revela alegría, pero la voz revela neutralidad/miedo, este detector de dos niveles detectaría una situación de nervios o estrés, cosa que usando solo un detector no habríamos capturado.

Con esto se persigue **profundizar en el estudio de tecnologías de detección de emociones** con miras a seguir trabajando en la variación dinámica del comportamiento de las aplicaciones en función de las emociones detectadas en el usuario, retomando así una línea de trabajo existente [4]. El objetivo este TFM es crear un sistema de detección de emociones más sofisticado que el creado en [4], utilizando para ello un sistema multimodal de dos niveles.

En la Figura 1 podemos ver un ejemplo esquemático de cómo sería un detector parcialmente multimodal para la expresión facial y la voz. En primer lugar, se seleccionan varios detectores de emociones basados en expresión facial y en la voz y se *agrupan*. Cuando se quieran analizar las emociones reflejadas en una expresión facial o en una voz hablando, se pasará la imagen o la pista de audio a cada detector del grupo que corresponda, emitiendo cada *detector individual* un resultado. Estos resultados, a su vez, se combinarán en uno solo, que constituirá el resultado global producido por ese grupo de detectores. De esta manera, se están validando de forma cruzada los resultados individuales, produciéndose así una *clasificación de emociones individual más sólida*. Esto constituye el primer nivel de la aplicación.

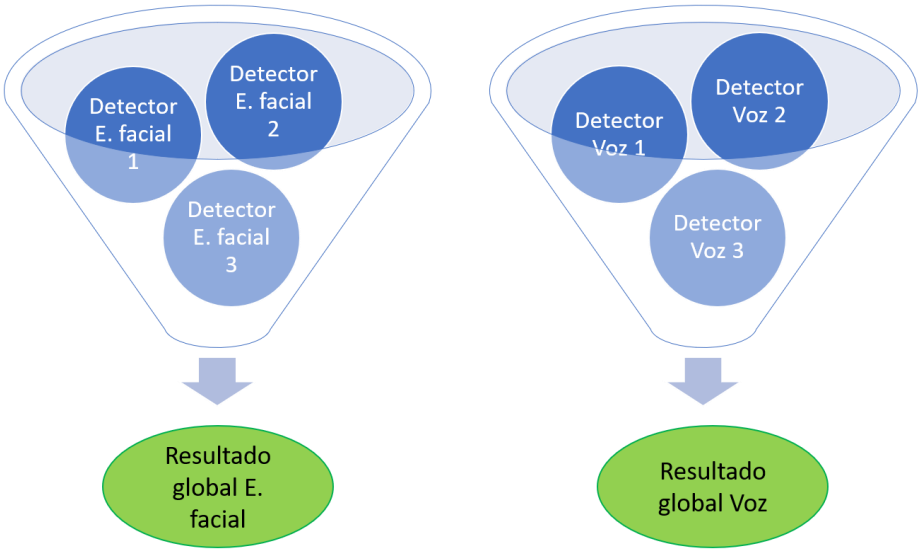

Figura 1. Sistemas de validación monocanal para expresión facial y la voz

En el segundo nivel, se fusionarán los resultados de cada detector multimodal individual, con el objetivo último de confirmar, ampliar y/o desmentir la clasificación obtenida a través de otros canales (Figura 2). En última instancia, esto supone una clasificación más fiel para los usuarios de dicho sistema y, al mismo tiempo, abre la puerta

a la detección de estados más complejos o aspectos de la persona estudiada, como su concentración, su nivel de estrés, etc.

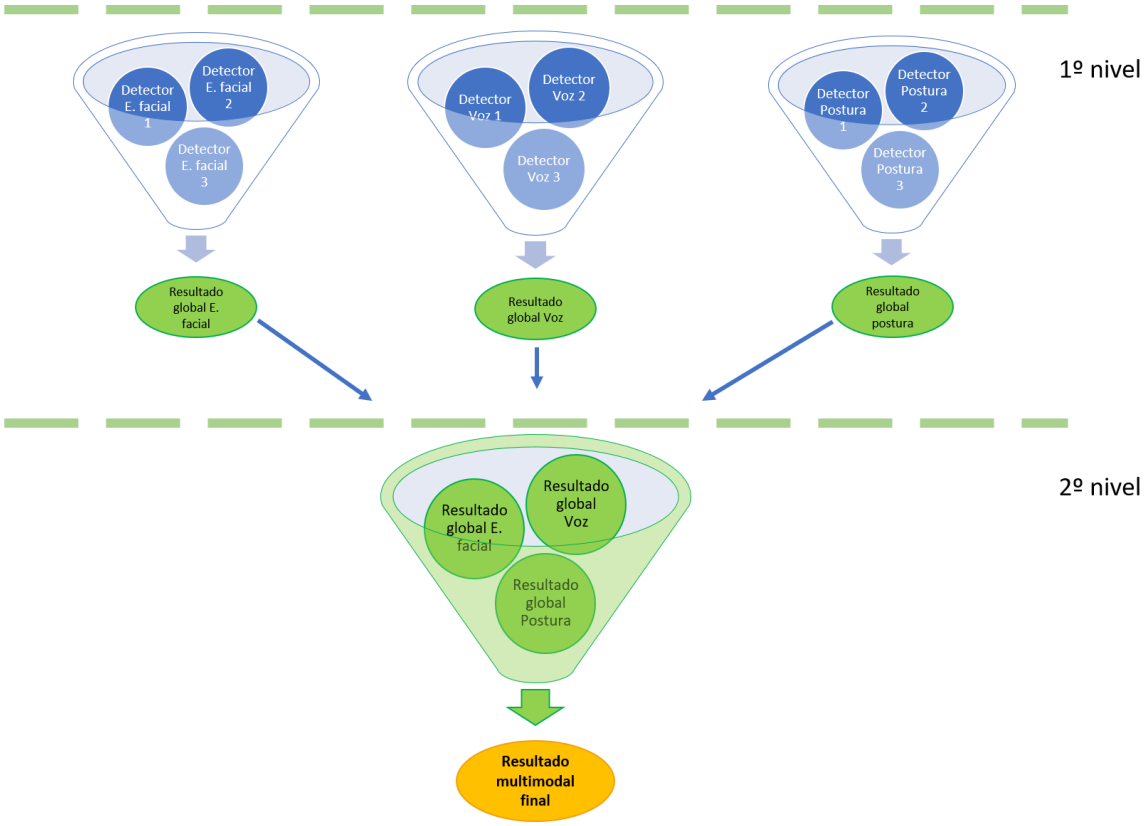

Figura 2. Fusión multimodal en segundo nivel

1.2 Objetivos

El **objetivo principal** de este TFM consiste en **crear un sistema de detección de emociones multimodal a dos niveles** que nos permita integrar los resultados parciales de distintos canales afectivos para, entre otras cosas, mejorar la clasificación, tanto individual como conjunta, de las emociones detectadas y ser capaces de detectar estados afectivos más complejos.

Para conseguir este objetivo, se plantean los siguientes *objetivos específicos*.

- **Realizar una revisión sistemática de la literatura**. Más allá de conocer los todos los detalles de todas las investigaciones realizadas en el campo de la Computación Afectiva en los últimos años, lo que se persigue es saber dónde se está aplicando más y cómo, de cara a emplear los esfuerzos de desarrollo de este trabajo de una manera o de otra.

- **Analizar las tecnologías más utilizadas y/o más avanzadas.** Fruto de la revisión sistemática obtendremos una lista de las tecnologías de detección de emociones que más se estén utilizando en proyectos de aplicación real.
- **Modelar de forma individual y conjunta distintos sistemas de detección de emociones**. Para poder combinar distintos detectores en un único sistema, es necesario modelar previamente cada detector de forma individual, considerando a su vez que esos detectores se alimentarán de servicios que generarán datos en formatos distintos.
- **Combinar servicios de detección de emociones del mismo tipo para obtener más precisión en los resultados**. La creación de un sistema multimodal que reciba entradas de múltiples tipos de servicios y los combine de la forma adecuada nos garantizará una mayor tasa de acierto.
- **Estudiar resultados de servicios combinados para detectar resultados incongruentes**. Dado que ese sistema multimodal considerará respuestas de varios sistemas a la vez, esto nos permitirá analizan incongruencias, emociones falsas y/o complejas y estados cognitivos imposibles de detectar usando los detectores por separado.
- **Definir un framework que permita la detección de emociones multimodal.** A partir de los resultados de los objetivos anteriores, se definirá la arquitectura del sistema y un framework que permita integrar la detección de emociones en otras aplicaciones.

El sistema se desarrollará en una plataforma en la nube para asegurar su mayor disponibilidad, haciendo uso de las tecnologías que se consideren más adecuadas.

1.3 Estructura de la memoria

Este documento queda dividido en seis capítulos, incluido el actual. A continuación, se muestra una breve descripción acerca del contenido de cada capítulo:

- **CAPÍTULO 1. Introducción**. Este capítulo contiene las motivaciones que han llevado a la realización de este trabajo, así como los objetivos perseguidos durante el desarrollo de este.
- **CAPÍTULO 2. Conceptos relacionados y estado del arte**. Este capítulo continúa el corpus de conocimiento iniciado ya en el Trabajo de Fin de Grado [4] que precede a este TFM. Se llevará a cabo una revisión sistemática de la literatura para conocer el estado del arte en el campo de la Computación Afectiva.

- **CAPÍTULO 3. Propuesta de solución**. En este capítulo se presenta la metodología de trabajo que se va a seguir para el desarrollo del prototipo, así como una justificación de la elección de una metodología sobre otra, junto con las tecnologías y herramientas que se van a utilizar.
- **CAPÍTULO 4. Desarrollo del sistema.** En este capítulo se describen detalladamente los *sprints* en los que ha quedado dividido el desarrollo del sistema. Cada sprint está dividido a su vez en tres apartados, a saber, planificación, desarrollo y revisión, en los que se presentan, respectivamente, las tareas a realizar en ese sprint, una breve explicación sobre la ejecución de las mismas y un análisis del producto obtenido al final del sprint.
- **CAPÍTULO 5. Descripción de la aplicación**. En este capítulo se hace una descripción final del producto obtenido al acabar todos los sprints. Se detallan los aspectos más destacados de la aplicación, como las condiciones necesarias para su despliegue, la forma de integrarla en otra aplicación, las aportaciones que supone su integración respecto a enfoques más clásicos, etc.
- **CAPÍTULO 6. Conclusiones y propuestas**. Por último, se hace una revisión del trabajo realizado, extrayéndose unas conclusiones relativas a los temas tratados a lo largo del trabajo, y se proponen una serie de aspectos en los que se podría trabajar en un futuro.

CAPÍTULO 2. CONCEPTOS RELACIONADOS Y ESTADO DEL ARTE

En este capítulo se detallará la tarea de investigación que se realizó con el fin de crear una base de conocimiento sólida sobre la que sustentar este trabajo. Se revisará la metodología propia de las revisiones sistemáticas, se adoptará una estrategia para realizarla y, posteriormente, se analizarán los datos obtenidos.

Este capítulo supone una continuación y ampliación de su homónimo en el Trabajo de Fin de Grado “Variación Dinámica del Comportamiento de las Aplicaciones en Función de las Emociones de Usuario” que supone el antecesor de este mismo TFM, por lo que se referenciará parte de su contenido a modo de enlace.

2.1 Computación afectiva

La Computación Afectiva, término introducido por primera vez en [1], se define como la «computación relacionada con, provocada por, o que influye en emociones». En otras palabras, cualquier forma de computación que tiene algo que ver con emociones de forma directa: las emociones no son algo separado de la interacción persona-ordenador, sino que forman parte de la misma. Por ejemplo, una página web que contuviese información sobre emociones o que contuviese una historia que provocase tristeza en el lector de la misma no se consideraría Computación Afectiva, mientras que una página que modificase su aspecto en base al estado anímico del usuario sí lo sería. [4]

A pesar de que la definición de Computación Afectiva pueda resultar un poco vaga, en la práctica se traduce, en la gran mayoría de los casos, en detectar las emociones del usuario y en usar esas emociones con la finalidad de mejorar su experiencia de usuario al interactuar con el sistema. Es por ello por lo que este trabajo se centra en la rama de la Computación Afectiva relativa a la detección de emociones.

Aunque la Computación Afectiva es una rama relativamente nueva, las emociones han sido un tema que lleva preocupando a filósofos y pensadores durante siglos. Su origen, su significado, su influencia, su transmisión: estos temas no son ninguna novedad en el panorama científico. En la actualidad, sabemos que las emociones se originan en el sistema límbico, una estructura del cerebro, situada debajo del córtex, a cargo de las emociones, la atención y la memoria [1]. Al sentimiento en sí que siente una persona en un momento dado se le denomina *estado afectivo*, mientras que la comunicación de dicho estado al exterior tiene lugar a través de la *expresión de emociones*. Dicha expresión tiene lugar a través de *respuestas físicas* del cuerpo humano: lenguaje corporal, ritmo cardiaco, expresiones faciales, tono de la voz, etc. Es por esto que merece la pena destacar que las distintas tecnologías de detección de emociones que van a utilizarse y a integrarse en este TFM van a trabajar detectando las *expresiones de las emociones*, intentando, a través de estas dilucidar el estado afectivo de la persona. Mientras que los ordenadores no sean capaces de mapear con una precisión absoluta respuestas fisiológicas a estados afectivos, no se podrá decir que un computador detecta estados afectivos.

Dado que, por el momento, un computador solo puede ver las emociones que provoquen alguna clase de respuesta externa en el cuerpo o que la persona revele durante la interacción con otra entidad, la detección se reduce a los siguientes medios [3]:

- Voz
- Expresiones faciales
- Lenguaje corporal y movimientos del cuerpo
- Respuestas fisiológicas
- Texto

De manera adicional, se puede usar información combinada de estos canales para obtener una mejor tasa de acierto en la detección de emociones. Esto se puede conseguir a través de *sistemas multimodales*. Este tipo de sistemas, que se presentarán con más detalle después de la revisión sistemática, son la piedra angular de este trabajo.

Aunque cada tipo de tecnología trabaja de una manera concreta, la mayoría de ellas comparte un núcleo común en lo que concierne a su funcionamiento. Esto se debe a que un detector de emociones como tal, es, fundamentalmente, un *clasificador automático*. La creación de un clasificador automático implica recoger información, extraer las características que resulten verdaderamente importantes para el objetivo que se persigue y finalmente, entrenar el modelo para reconocer y clasificar ciertos patrones [5].

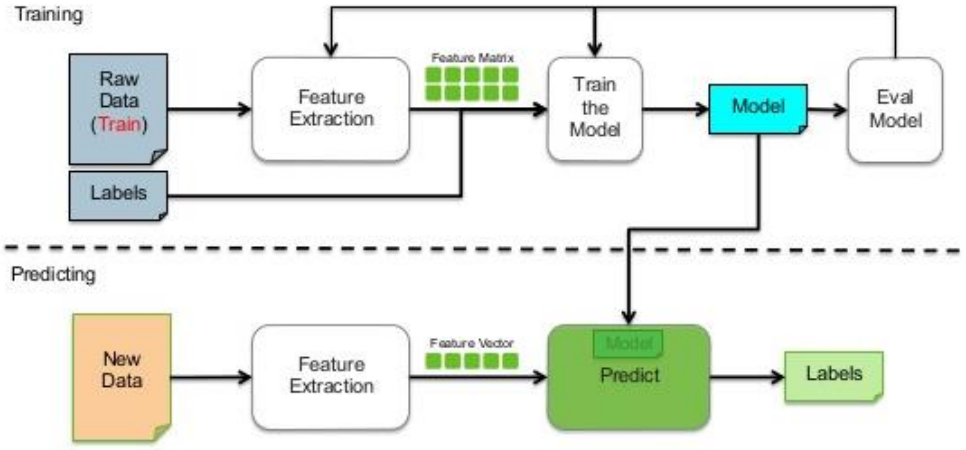

Figura 3. Flujo de trabajo de aprendizaje supervisado

Posteriormente, se utilizará el modelo entrenado para clasificar nuevos datos. En la Figura 3 [5] podemos ver todo el flujo de entrenamiento y posterior uso de un clasificador. Por ejemplo, supongamos que se quisiera construir un modelo que fuese capaz de identificar alegría y tristeza en expresiones faciales. Para entrenar este modelo, habría que suministrarle imágenes ya clasificadas: las imágenes que muestren gente sonriente deben ir clasificadas con la etiqueta “Alegría”, mientras que las que muestren a gente de expresión alicaída y ceño fruncido deben ir clasificadas con la etiqueta “Tristeza”. Después de llevar a cabo este entrenamiento, cuando el modelo recibiese una imagen de una persona sonriendo, este identificaría la emoción expresada como “Alegría”, mientras que ante una imagen de una persona con expresión alicaída devolvería “Tristeza” como resultado [4].

Para la justificación de este trabajo, decidió llevarse a cabo una revisión sistemática de la literatura que permitiera comprobar, de primera mano, el estado del arte de la aplicación de la Computación Afectiva. Dicha revisión se presenta en la siguiente sección.

2.2 Revisión sistemática de la literatura

Si bien durante la elaboración del TFG [4] que constituye la precuela de este trabajo se construyó una base de conocimiento considerable, esta resulta insuficiente de cara a una propuesta más sólida que inicie la línea de investigación de una tesis doctoral. Es por esto que decidió llevarse a cabo una *revisión sistemática de la literatura*, centrada sobre todo en la *aplicación de la Computación Afectiva*.

Una revisión sistemática de la literatura [6] es un tipo de revisión de la literatura que persigue identificar, evaluar e interpretar toda la investigación disponible y relevante sobre un tema en particular. Dado que una revisión sistemática revisa estudios conocidos como «primarios», se las considera un tipo de estudio «secundario». Gracias a este análisis podemos localizar lagunas en la investigación, detectar sesgos en las publicaciones, observar patrones en los artículos publicados, etc. La preparación de este tipo de estudios se puede dividir en *tres fases*.

- **Identificar la necesidad de una revisión.** Hemos de comprobar la existencia de otras revisiones sistemáticas, si existieran, para no repetir exactamente el mismo trabajo y, si las hubiere, considerar cómo se realizaron en su momento, estudiando posibles nuevos enfoques, criterios de aceptación, estrategias de síntesis, etc.
- **Plantear las preguntas de investigación**. Una de las cosas que diferencia a una revisión sistemática de un proceso cualquiera de búsqueda de información es su formalidad. En lugar de buscar información bajo demanda para explorar un terreno, se plantean una o varias preguntas que constituirán la «hipótesis» que se pretende demostrar o rebatir con dicha revisión.
- **Definir la estrategia de búsqueda**. Por último, hemos de establecer qué criterios usaremos para aceptar o rechazar un artículo en la revisión, qué tipo de publicación aceptaremos, qué bases de datos consultaremos, etc.

En las siguientes subsecciones de este capítulo se identificará la necesidad de una revisión sistemática, se establecerán las preguntas de investigación que servirán como base al proceso, se detallará la estrategia de búsqueda a seguir y, por último, se presentarán los resultados.

2.2.1 Necesidad de una revisión sistemática

En el TFG [4] que abrió la línea de trabajo que hoy continúa este TFM, se realizó una revisión de la literatura para conocer la rama de la Computación Afectiva, las tecnologías que se utilizaban ese sector, los usos y aplicaciones más comunes de esta rama,

etc. Si bien ese sondeo devolvió una gran cantidad de información muy valiosa, quedaba lejos de ser un proceso formal y definido. Es por ello que se decidió cubrir esa carencia en este trabajo.

Como parte de la planificación de la revisión, se realizó una búsqueda en diversas bases de datos de artículos científicos, con el fin de comprobar si ya existían revisiones o mapeos sistemáticos de la literatura sobre el tema a tratar, a saber, *Computación Afectiva y su aplicación*. Aunque se encontraron muchas revisiones de la Computación Afectiva como disciplina (*reviews*), todas ellas se centraban en hacer un análisis casi *histórico*, revisando tipos de detecciones de emociones, teorías de la emoción más antiguas sobre las que se construía la detección de emociones actual, tecnologías usadas en esa detección, etc. A pesar del estallido de actividad que se ha dado en los últimos años en esta materia, y de la gran cantidad de proyectos y aplicaciones reales que existen actualmente, no se ha llevado a cabo una revisión rigurosa de la bibliografía existente.

En este apartado se pretende corregir esa deficiencia, aplicando la metodología de investigación propia de una *revisión sistemática de la literatura* (Systematic Literature Review, *SLR*) para conocer el estado actual de la disciplina comentada anteriormente, conocer a los autores más destacados en ese campo, así como los proyectos realizados más importantes, identificar nichos y desiertos de oportunidades y sentar la base científica de un trabajo de investigación que se continuará en una tesis doctoral.

2.2.2 Planificación de la revisión

Como se introdujo anteriormente, para llevar a cabo la revisión sistemática de la literatura, hemos de seguir los siguientes pasos:

- Selección de las preguntas de investigación.
- Definición de la estrategia de búsqueda.
- Selección de estudios primarios.
- Evaluación de los estudios recogidos.

En las siguientes páginas se detallará en qué ha consistido la realización de esos distintos pasos. Si bien en [6] se hace referencia a una fase de difusión, sobre esta fase no se hablará explícitamente, puesto que la difusión del estudio consiste en la publicación de este mismo trabajo.

2.2.3 Preguntas de investigación

La parte más importante de una revisión sistemática es la fase de selección de las preguntas de investigación, ya que estas dirigirán el proceso de revisión: los estudios primarios que se busquen habrán de abordar el tema de la pregunta, los datos extraídos de los mismos han de ser aquellos necesarios para responder la pregunta y la síntesis de estos datos ha de ser capaz de dar una respuesta a la misma.

En este caso, el objetivo de la revisión es conocer, no solo qué aplicaciones de la Computación Afectiva se han realizado en otros campos, sino también cómo han sido éstas. Así, una forma de plantear esta pregunta podría ser la siguiente:

¿Cuál es el estado del arte sobre la aplicación de la Computación Afectiva en otros campos, y qué tipo de elementos se han empleado en esta aplicación?

Dada la amplitud de la pregunta, se ha dividido en preguntas más específicas de manera que sea más fácil de abordar. Así, la pregunta de investigación queda dividida en las siguientes subpreguntas:

- RQ0.1. ¿Quiénes son los autores más destacados?
- RQ0.2. ¿Cuáles son las revistas o conferencias más importantes?
- RQ1. ¿En qué campos se han producido más aplicaciones de la Computación Afectiva?
- RQ2. ¿Qué aspectos se pretenden mejorar o corregir aplicando la Computación Afectiva?
- RQ3. ¿Qué elementos software o hardware se emplean en esta aplicación?
- RQ4. ¿Qué características tienen los participantes que responden mejor a la aplicación de la Computación Afectiva?
- RQ5. ¿Son comunes los juegos serios o las técnicas de gamificación en este tipo de proyectos?
- RQ6. ¿Cómo se evalúa el impacto que ha tenido la aplicación de la Computación Afectiva en los participantes?

A continuación, se detallará la estrategia que se seguirá para intentar dar respuesta a estas preguntas.

2.2.4 Estrategias de búsqueda

En este apartado se concretarán las fuentes de información que se utilizarán durante el proceso de revisión, así como los términos de búsqueda que se emplearán para explorar las mismas en busca de bibliografía. Se definirán también los criterios a seguir para aceptar o rechazar una publicación.

Términos de búsqueda

Para registrar las bases de datos de bibliografía es necesario introducir una serie de términos clave en sus filtros de búsqueda. En este caso concreto, es interesante realizar búsquedas acerca de proyectos en los que se aplique la Computación Afectiva. Más concretamente, es interesante conocer proyectos en los que se aplique la *detección de emociones*, preferiblemente sin sensores y de forma no intrusiva, aunque tampoco se rechacen aquellos en los que se utilice algún tipo de sensor conectado a un microcontrolador como Arduino o Raspberry.

Aunque estamos interesados en la aplicación en general de la detección de emociones, resulta más valioso, de cara a este trabajo, investigar sobre aquellos proyectos que estén dirigidos a *personas con necesidades especiales*, sobre todo si se trata de niños, puesto que esta fue la línea que se abrió durante el TFG.

Dada la popularidad de las técnicas de gamificación y de los juegos cuya función principal no es lúdica («juegos serios»), es interesante también ver si estos tienen presencia en conjunto con la Computación Afectiva.

Por tanto, es necesario definir:

- *palabras clave*, así como sinónimos y/o términos alternativos;
- *cadenas de búsqueda* combinando esas palabras clave.

Las palabras clave consideradas para esta revisión son:

- *affective computing, emotion, sentiment, mood, detection, analysis*;
- *service*, API, *sensor, detector, arduino, raspberry*;
- *special, need, limitation*;
- *gamification, game, serious, serious game, gamified*;
- *measurement, evaluation*.

A partir de las palabras clave se construyen las cadenas de búsqueda que ayudarán a determinar el conjunto inicial de estudios que se tendrán en cuenta en esta revisión.

- "*affective computing*" AND ("*emotion*" OR "*sentiment*" OR "*mood*") AND ("*detection*" OR "*recognition*")
- "*affective computing*" AND ("*service*" OR "API" OR "*sensor*" OR "*detector*" OR "*arduino*" OR "*raspberry*") AND ("*emotion-based system*" OR "*emotion-based application*" OR ("*technology*" AND "*emotions*"))
- "*affective computing*" AND ("*emotion*" OR "*sentiment*" OR "*mood*") AND ("*detection*" OR "*recognition*") AND (("*special*" OR "*need*" OR "*limitation*") OR ("*gamification*" OR "*game*" OR "*serious game*" OR "*gamified*"))
- "*affective computing*" AND ("*emotion*" OR "*sentiment*" OR "*mood*") AND ("*detection*" OR "*recognition*") AND (("*special*" OR "*need*" OR "*limitation*") OR ("*gamification*" OR "*game*" OR "*serious game*" OR "*gamified*")) AND ("*evaluation*" OR "*measurement*" OR "*assessment*")

Así, nos queda una cadena de búsqueda algo más genérica, una segunda cadena centrada en la tecnología usada y, por último, dos cadenas más enfocadas en el uso de Computación Afectiva aplicada a necesidades especiales y a procesos *gamificados*, considerando también que los artículos que se centren en ese tema incluyan algún proceso de evaluación.

Durante el proceso de revisión, esas cadenas de búsqueda se adaptarán a la sintaxis que exija cada una de las bases de datos. Los resultados producidos por cada base al procesar esas cadenas de búsqueda serán filtrados, de manera que solo se tendrán en cuenta finalmente aquellos que posean ciertas características y cumplan una serie de criterios previamente definidos.

Recursos usados para la búsqueda de información

En la Tabla 1 se pueden apreciar los datos asociados a la búsqueda de bibliografía. La búsqueda de bibliografía se terminó restringiendo al catálogo completo de la ACM Digital Library, con 2,820,514 registros. Se ha limitado la búsqueda a aquellos trabajos escritos en inglés durante los últimos diez años, y el tipo de publicación se ha limitado a artículos en revistas, en congresos y conferencias y talleres de trabajo.

2.2.5 Criterios de selección

A pesar del gran número de artículos que se puede conseguir durante el proceso de búsqueda, no todos supondrán una aportación a esta revisión y, por tanto, habrá que desecharlos. En este apartado se presentan los criterios que determinarán cuándo deberá considerarse un artículo y cuándo no.

Tabla 1. Recursos utilizados

Bases de datos	• ACM Digital Library
Tipos de publicaciones	• Artículos en revistas de investigación • Congresos y conferencias • Talleres
Búsqueda aplicada a	• Título • Resumen • Palabras clave
Lenguaje	• Inglés
Periodo de publicación	• Desde enero 2011 hasta enero 2018

Criterios de inclusión

- El artículo está contenido en las bases de datos especificadas en la Tabla 1 y es un tipo de artículo de identificado en esa tabla.
- El artículo está centrado en la temática de la Computación Afectiva y en la aplicación real de ésta.
- Si se encontrasen distintas versiones de un artículo con fechas distintas, se tendrá en cuenta solo el más reciente.
- Si un documento constituye un compendio de trabajos, cada uno se considerará como un trabajo individual.

Criterios de exclusión

- El artículo comenta el potencial de la Computación Afectiva, pero sin presentar ninguna aplicación real de la misma.
- El artículo está centrado en la propuesta de un algoritmo o metodología que mejore la tasa de acierto de un detector de emociones o proponga una nueva forma de detectar emociones sin llegar a aplicarla.

- El artículo se dedica a presentar una o varias tecnologías de detección de emociones, sin aplicarlas a ningún proyecto concreto o las estudia fuera de cualquier contexto.
- El artículo habla sobre evaluaciones relacionadas con la Computación Afectiva pero no realiza ninguna.
- El documento está en formato de presentación PowerPoint.
- El artículo no está escrito en inglés.
- Cuando el trabajo haya aparecido en bases de datos distintas se contabilizará solo una vez.

2.2.6 Selección de trabajos relevantes

En la Figura 4 podemos ver el flujo de tareas que se llevó a cabo para seleccionar los artículos que se considerarían en la revisión sistemática. Para iniciar la búsqueda de bibliografía, se introdujeron las cadenas de búsqueda en el buscador de ACM Digital Library [7]. En el caso de este buscador, hemos de replicar cada cadena de búsqueda y encabezar cada copia con los prefijos «acmdlTitle» y «recordAbstract» para que el buscador compruebe explícitamente el título y el resumen de cada artículo en cada búsqueda. Para realizar una primera selección, se analizaron el título y resumen de los artículos que la búsqueda devolvió, aplicando los criterios de selección definidos anteriormente para desechar artículos cuyo tema diverja demasiado del nuestro.

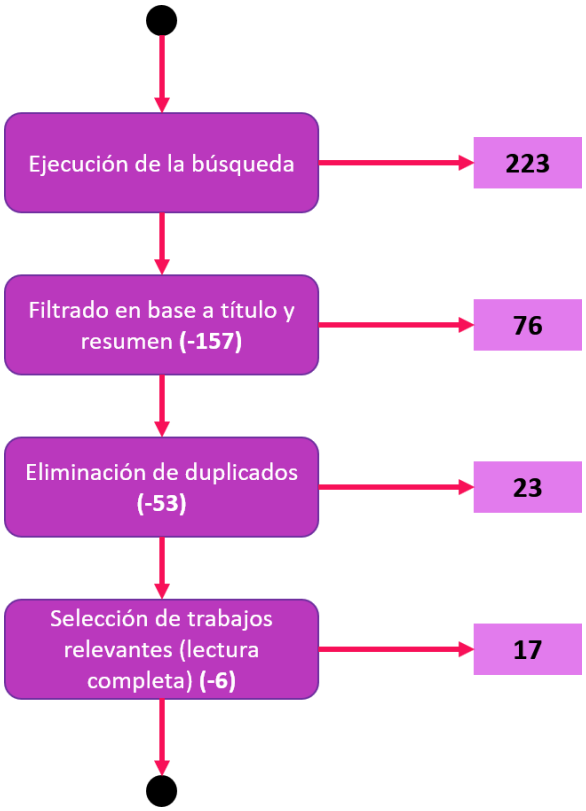

Figura 4. Proceso de selección de artículos

Hecho ese primer filtrado, se eliminaron los *artículos duplicados*, esto es, que han aparecido en búsquedas distintas, y, tras una lectura completa de los artículos elegidos, se desechó una última tanda de trabajos que no ofrecían el contenido que su título y resumen parecían prometer. Tras un análisis de estas lecturas restantes, se recopilaron los datos que se exponen en la siguiente sección.

2.2.7 Resultados

Autores y revistas importantes (RQ0.1, RQ0.2)

A pesar del reducido número de artículos que han quedado al final de la revisión, ciertos autores han tenido mayor presencia que otros (Figura 5).

- *Michael Oehl*, German Aerospace Center (DLR), Institute for Transportation Systems – HMI Group, Alemania
  - 2 publicaciones
- *Felix W. Siebert*, Technische Universität Berlin, Alemania
  - 2 publicaciones

- *Tessa-Karina Tews*, Leuphana University Lüneburg, Alemania
    - 2 publicaciones
- *Rainer Höger*, Institut für experimentelle Wirtschaftspsychologie, Alemania
    - 2 publicaciones

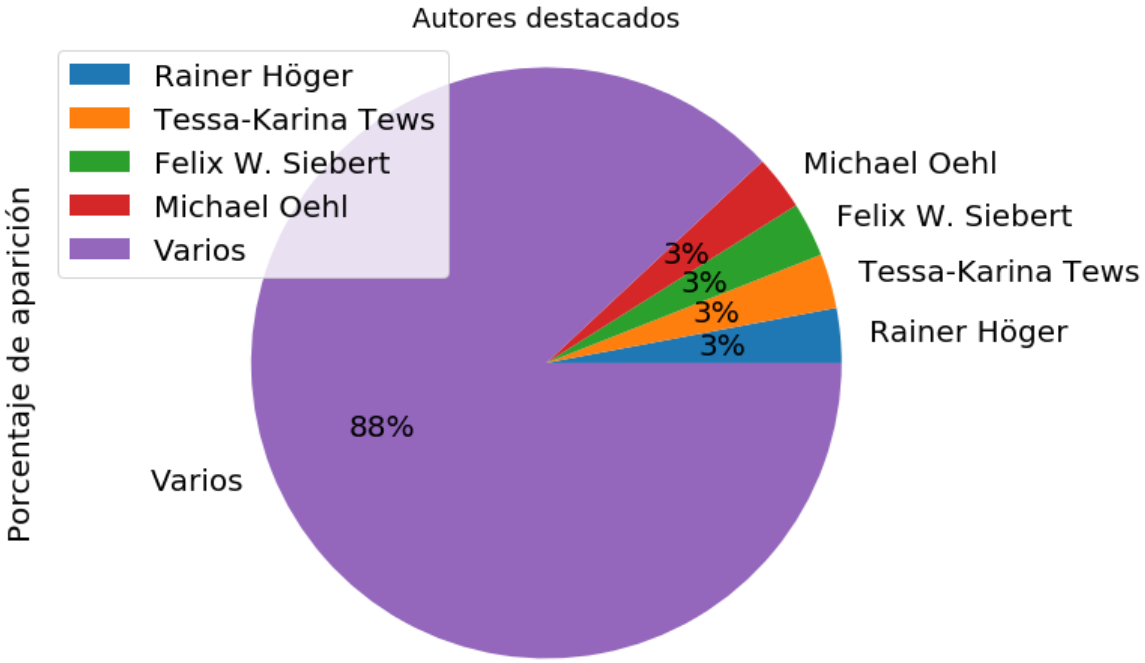

Figura 5. Autores destacados

En cuestión de revistas, ha aparecido una mayor diversidad. Aunque ya se sabía que la organización «*Institute of Electrical and Electronics Engineers*» disponía de una revista centrada exclusivamente en la Computación Afectiva, *IEEE Transactions on Affective Computing*, la revisión ha descubierto que, en cuestión de aplicación, no está tan a la cabeza como podría pensarse. En la Figura 6 podemos ver que, si bien IEEE es el editor que más presencia ha tenido en la búsqueda, está seguido muy de cerca por otros como Springer y Hindawi.

- IEEE
    - 4 publicaciones
- Springer
    - 3 publicaciones
- Hindawi
    - 2 publicaciones
- IGI Global
    - 2 publicaciones

- ACM
    - 1 publicaciones
- Elvesier
    - 1 publicaciones
- BCS HCI
    - 1 publicaciones
- ScitePress
    - 1 publicaciones

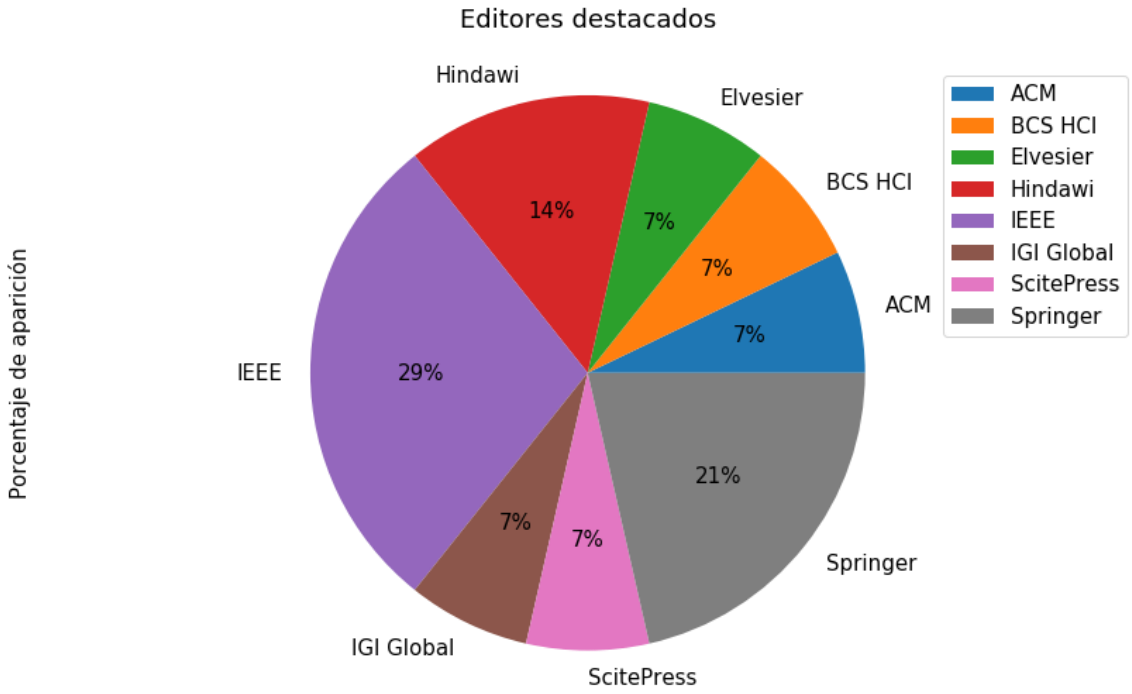

Figura 6. Editores destacados

Campos de mayor aplicación (RQ1)

Otra de las preguntas de la revisión hacía alusión a los campos de conocimiento en los cuales la Computación Afectiva tiene mayor aplicación. En este caso, la **Psicología** es la rama omnipresente en todos los artículos y experimentos. Incluso cuando se trata de, por ejemplo, un experimento para mejorar la concentración de los estudiantes o para analizar el comportamiento de un comprador en un centro comercial, es la psicología la disciplina que subyace todo el tiempo, al menos cuando se trata de la aplicación de la Computación Afectiva. No obstante, para desgranar algo más la clasificación se ha considerado que un artículo está en el campo de la Psicología como tal cuando trata temas como la depresión o la carga cognitiva de forma directa. Teniendo esto en cuenta, la clasificación queda así:

- *Psicología*
  - 4 publicaciones
- *Experiencia de usuario (UX)*
  - 4 publicaciones
- *Educación*
  - 3 publicaciones
- *Marketing*
  - 3 publicaciones
- *Medicina*
  - 2 publicaciones

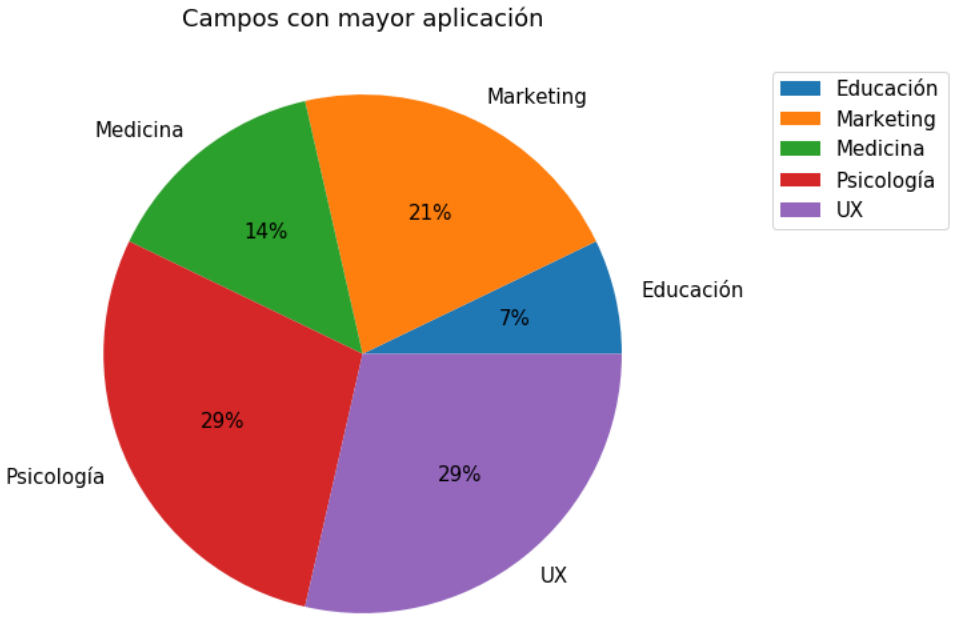

Figura 7. Campos con mayor aplicación

En la Figura 7 vemos un diagrama de sectores representando los números de la lista anterior. La **Psicología** es la disciplina en la que se enmarcan más artículos, seguida por el **Marketing** y los estudios de la **Experiencia de Usuario** (UX), ambos campos muy relacionados, y la **Educación**.

Aspectos para mejorar en los experimentos (RQ2)

Durante el proceso de revisión de los artículos seleccionados se apreciaron ciertos patrones en lo que al aspecto que se pretende mejorar con el estudio se refiere. Estos aspectos son:

- *Atención, concentración y carga cognitiva.* Un considerable número de artículos giraba en torno al estudio de la atención, la concentración y la carga cognitiva de los usuarios, ya fueran estudiantes en clase, personas conduciendo o usuarios intentando entender una página web.
- *Experiencia de usuario.* Otro aspecto también presente, si bien algo menos que el anterior, es la experiencia de usuario. En algunos de los artículos se ha analizado cómo influye en el usuario o en sus decisiones la aplicación o el sistema que están usando, normalmente con miras a mejorarlo. Cabe mencionar que la palabra «usuario» cambia por «cliente» en algunos estudios, estudios en los que se comprueba cómo reacciona un cliente a la disposición de las tiendas en un centro comercial o una máquina de autoservicio.
- *Autocontrol.* La idea del *emotion awareness*, esto es, ser consciente de las emociones de uno mismo, es relativamente nueva en el campo de la Computación Afectiva, y ha aparecido también durante la revisión en artículos de los últimos dos años. Por el momento es un concepto que se suele presentarse asociado a Trastornos del Espectro Autista (TEA), puesto que las personas que padecen este trastorno tienen problemas para gestionar sus estímulos y sus reacciones a ellos. No obstante, en el campo de la Psicología se reconocen los beneficios de ejercitar esa *autoconciencia* de las propias emociones [8].
- *Calidad de vida.* Tienen también gran presencia los estudios en los que se busca mejorar la calidad de vida de las personas. En esta revisión han aparecido artículos en los que se proponen herramientas para asistir en la detección de la **depresión** y durante procesos de rehabilitación y terapia.

De esta manera, podemos ver que la lista anterior de aspectos a mejorar o a corregir se parece mucho a la clasificación previa de los campos que tienen mayor aplicación.

Elementos hardware o software utilizados (RQ3)

Aun cuando los estudios revisados son bastante actuales, son pocos los que hacen uso de tecnologías más modernas o flexibles, como los microcontroladores o los sensores de dispositivos móviles, para llevar a cabo la detección de emociones. Siguen predominando la *cámara web*, el *micrófono,* el *teclado* y el *ratón* como periféricos para detectar emociones. Solo en tres artículos se han usado dispositivos que no estuvieran en la lista anterior, a saber, wifi y lectores NFC integrados en el móvil, un casco para la detección de la actividad cerebral y una pulsera para medir la actividad electrodérmica de los sujetos. Además de estos medios electrónicos, siguen usándose herramientas de *autoevaluación*, mediante las cuales los propios usuarios dejan constancia de cómo se sentían durante la evaluación. Es cierto que estas herramientas permiten crear un conjunto de datos contra los que contrastar la clasificación de los detectores de emociones, pero el

uso de herramientas como, por ejemplo, el Self-Assessment Manikin, viene desaconsejándose desde hace tiempo, debido al sesgo que añade a la evaluación la propia subjetividad de los participantes.

El Self-Assessment Manikin (Maniquí de Autoevaluación) es un formulario que usa distintas representaciones gráficas de un maniquí para que los participantes de un experimento evalúen cómo se sentían durante el mismo. En la Figura 8 podemos ver un ejemplo de ese formulario. La primera fila se utiliza para expresar la *valencia* del sentimiento (negativo o positivo), la *segunda* fila se utiliza para expresar la excitación (cuán fuerte o débil es ese sentimiento) y la última expresa la *dominancia* (si ese sentimiento nos hace sentir más dominantes o más pasivos).

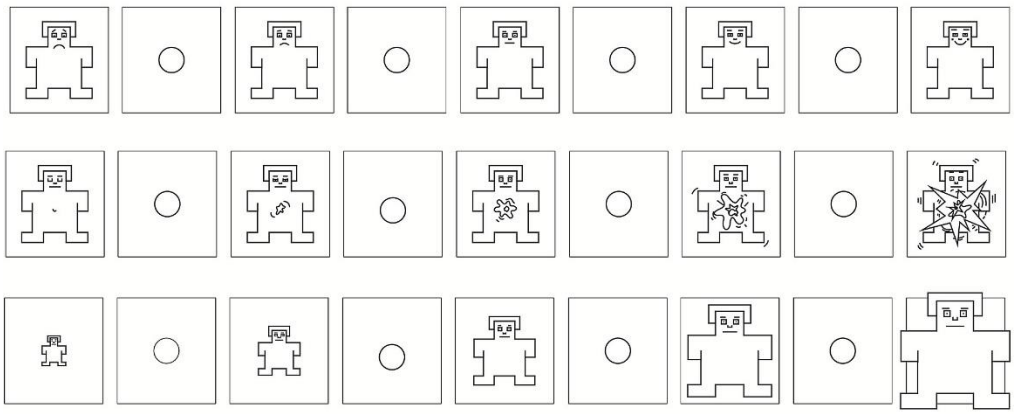

Figura 8. Self-Assessment Manikin

Características de los participantes (RQ4)

En lo que respecta a esta pregunta no se ha encontrado ninguna respuesta concluyente. Antes de empezar la revisión se tenía la impresión de que quizás, de cara a las evaluaciones con usuarios, se reclutase a gente con algún perfil concreto, o que ciertas personas respondiesen a la integración de elementos de la Computación Afectiva mejor que otras. No obstante, no se ha encontrado ninguna especie de patrón o característica común.

Si merece la pena destacar que la mayoría de los grupos de participantes tiene una edad media que se sitúa en torno a los *30 años*. Además, no se dan detalles sobre los medios utilizados para reclutar a los participantes, por lo que la sensación final es que los participantes de los experimentos son estudiantes universitarios que posiblemente han recibido clase de los autores de los estudios. De ser este el caso, estaríamos ante un *sesgo* considerable, puesto que quedan fuera del estudio personas de otros ámbitos socioeconómicos (personas que no han estudiado en la universidad), de otras disciplinas (personas de estudios distintos a los de los autores y participantes del experimento), etc.,

lo que, a su vez, daría como fruto unos resultados que no son extrapolables al resto de población de la que se ha extraído la muestra.

Juegos serios y gamificación (RQ5)

Los *juegos serios* son aquellos juegos hechos con una finalidad que no es totalmente lúdica. Por ejemplo, un juego a través del cual se aprende un idioma no es un simple juego, sino que es un juego serio. Este tipo de juegos aprovecha los beneficios de estos para invertirlos en mejorar procesos de aprendizaje.

Por otro lado, la *gamificación* se define como la aplicación de mecanismos propios de los juegos a otros entornos. La gamificación ha demostrado ser beneficiosa en muchos casos. Mayor *motivación*, mayor *participación*, mejores *experiencias*: todos estos son beneficios de aplicar los principios de la gamificación a entornos profesionales y educativos.

Con un solo vistazo podemos ver que estos dos conceptos podrían encajar muy bien con la Computación Afectiva. Detección de emociones mientras se está usando un juego serio (de hecho, este fue el trabajo desarrollado en el TFG [4]), uso de agentes virtuales que simulen tener emociones para simpatizar con usuarios de aplicaciones *gamificadas*, etc. Sin embargo, solo 2 artículos de los 17 mencionaban, en este caso, el uso de juegos serios con una detección de emociones integrada. La gamificación ni siquiera ha salido a relucir en los artículos revisados. Es más, de entre los 157 artículos desechados, solo en 3 se planteaba la relación entre la Computación Afectiva y los juegos, o se estudiaba su posible integración.

En contra de las expectativas que se tenían en un principio, no se han hecho demasiados avances en lo que se refiere a la integración de la Computación Afectiva y los juegos serios.

Evaluaciones de experimentos (RQ6)

Una de las conclusiones más llamativas ha sido la respuesta a esta pregunta. De un conjunto de estudios relacionados con la aplicación de la Computación Afectiva y que persiguen, presuntamente, implantar nuevas formas de trabajar o de hacer ciertas cosas, se esperaba un proceso de evaluación mucho más exhaustivo con muchos más usuarios.

Son varios los estudios que realizan los experimentos usando datos de bases de datos o recopilados por ellos mismos a lo largo del tiempo, sin contrastar los resultados con datos «espontáneos». Por ejemplo, dos de los artículos seleccionados analizaban cómo detectar la depresión en base a información afectiva, para lo cual utilizaban datos de una base de datos con información de pacientes con depresión [9]. Para comprobar si el clasificador

propuesto funciona, utilizan datos de esa misma base, pero de gente sana, en lugar de probarlo con casos reales actuales.

Por otro lado, las evaluaciones que utilizan usuarios reales son muy variables en lo que se refiere a número de participantes. En la Figura 9 podemos ver un histograma en el que se representa cómo está distribuido el tamaño de los grupos de evaluación de cada estudio. Por un lado, tenemos cuatro grupos de evaluación que rondan los 60 miembros. Dos de esos grupos utilizan datos de 60 personas de una base de datos, mientras que los otros dos son realmente el mismo grupo de 59 personas (los autores de los dos artículos son los mismos, y utilizan el mismo grupo para hacer dos pruebas distintas, aunque muy parecidas). En el otro lado del gráfico podemos ver que, si bien hay tres grupos con un número de participantes que oscila entre el 24 y el 36, la gran mayoría se encuentra entre el 0 y el 18, habiendo una mayor tendencia hacia el cero.

En lo que respecta al análisis de los datos recopilados, los resultados también han sido poco esperados. No todos los estudios utilizan técnicas de análisis estadístico, sino que más bien se limitan a hacer observaciones sobre los datos y algunas de sus métricas, como la media, la desviación estándar, etc. Dentro de los que si hacen un análisis exhaustivo ha aparecido varias veces la *prueba T de Student*, muy recomendable para muestras pequeñas, como es el caso. Aunque de forma anecdótica, también aparece la prueba de Chi cuadrada y el coeficiente de determinación, e incluso en dos artículos se utiliza una combinación de varias pruebas estadísticas.

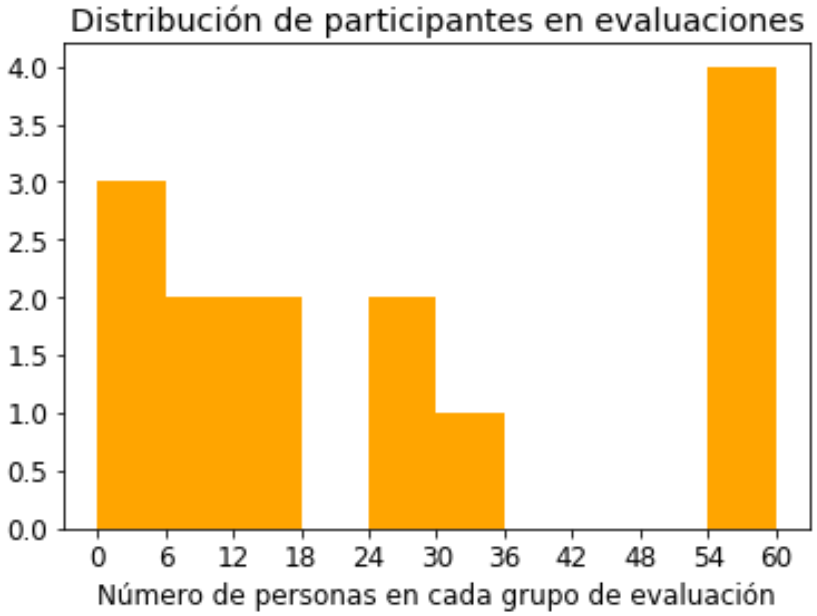

Figura 9. Histograma de número de participantes en experimentos

Conclusiones

Si bien se han considerado 17 artículos al final, descartados han quedado 157. Esto significa que, de 174 artículos, en un periodo de siete años, solo el **9.8%** trata de la *aplicación* de la Computación Afectiva. El **90.2%** restante se dedica, mayoritariamente, al *desarrollo de clasificadores de emociones*, probando nuevas técnicas de aprendizaje automático, considerando nuevas características que puedan revelar más información que otras, etc.

En general, la comunidad de investigadores está dejando de lado la posible aplicación de la Computación Afectiva. El *boom* del *Big Data* y el *Machine Learning* se mantiene presente en esa disciplina, y es por eso que la mayoría de los trabajos giran en torno a esos dos ejes. Desde un punto de vista empresarial, si bien la aplicación de la Computación Afectiva es un nicho económico que pocas empresas están explotando, la mayoría de implicadas están trabajando en la creación de mejores detectores, mejor *hardware*, etc., dejando una vez más, de lado, la aplicación.

Por último, cabe mencionar que la *detección multimodal* está a la orden del día en cuestión de aplicación. De los trabajos seleccionados, son muchos los que utilizan varios canales de entrada para sus detectores de emociones. La detección multimodal se conoce desde que se conoce la detección automática de emociones y es un hecho comprobado y reconocido que esta nos aporta una mayor tasa de acierto, mayor confianza, más información para entender las circunstancias de un usuario, etc. No obstante, usar una detección multimodal implica mayor complejidad, problemática, e incluso coste para los encargados de desarrollar el proyecto, por lo que suele dejarse para trabajo futuro. Siguiendo esta línea, es debido a esa complejidad y a esa dependencia de otros canales de información que los trabajos de mejora de los clasificadores actuales se han limitado a detectores de emociones *monocanales*.

En base a estas observaciones, tanto de los artículos descartados como de los seleccionados para la revisión, podemos concluir esta revisión sistemática sabiendo que:

- Se reconocen las ventajas de integrar las emociones de los usuarios en las aplicaciones que utilizan, pero no se están haciendo proyectos de aplicación *per se* (en comparación a los proyectos que se están realizando para mejorar los detectores).
- En los proyectos de aplicación predomina la *detección multimodal*. Incluso cuando esta pueda ser un poco tosca, sus numerosos beneficios la convierten en la mejor opción a la hora de detectar emociones.
- La *Psicología* en el principal campo de aplicación de la Computación Afectiva. Reducir carga cognitiva, optimizar procesos de aprendizaje,

desarrollar autoconsciencia de las propias emociones: estos son algunos de los trabajos que se han realizado durante los últimos años.

- La *voz* se encuentra casi desplazando a la *expresión facial* en el ranking de canales afectivos más utilizados, y señales fisiológicas como la actividad electrodérmica de la piel y la actividad eléctrica del cerebro han cobrado mayor presencia.
- El uso de detectores de emociones comerciales se ha visto muy reducido. La mayoría de trabajos disponen de datos recopilados por ellos mismos, aunque también extraídos de bases de datos de terceros, que utilizan para entrenar sus propios clasificadores. Además, el uso de señales fisiológicas, más frecuente que hace algunos años, hace innecesario el uso de esos clasificadores comerciales.
- En paralelo al punto anterior, la postura de las empresas que ofrecen detectores de emociones se ha vuelto mucho más *comercial*, haciendo que licencias que antes estuviesen disponibles para investigadores y/o estudiantes hayan dejado de estarlo.

Todo esto nos permite identificar una laguna que este TFM pretende empezar a cubrir. En las siguientes subsecciones distinguiremos algunos de los puntos clave de esa propuesta y en el capítulo siguiente se presentará tanto la idea como la planificación de su desarrollo.

2.3 Sistemas multimodales

La idea de detectar emociones de forma automática tiene la misma antigüedad que la Computación Afectiva. Cuando esta salió a escena, fueron muchos los estudios que surgieron explorando los distintos medios a través de los cuales podía obtenerse información afectiva. Los primeros canales en ser explotados fueron la expresión facial y la voz. En poco tiempo, otras vías como la expresión corporal, señales fisiológicas e incluso otras formas de interacción, como el texto escrito empezaron a considerarse.

No obstante, cuando aparecieron los primeros estudios sobre detección de emociones también apareció, de forma casi simultánea, la idea de la *multimodalidad*. La premisa de este concepto era sencilla: si se buscan emociones en un único canal (la cara, el cuerpo, la voz, etc.), es probable que se esté perdiendo información, ya sea porque el canal no es fiable, porque el sujeto medido oculta sus emociones reales, etc. Esta pérdida de información se traduce en una peor clasificación de las emociones. Para corregir ese problema, se *detectarán y estudiarán las emociones de varios canales distintos a la vez*, de manera que puedan realizarse detecciones más sólidas, contrastar detecciones entre los detectores de forma cruzada, etc.

En 1997, [10] dio el pistoletazo de salida al tema de la multimodalidad, estudiando cómo se podía detectar la emoción en un canal (la cara) usando como apoyo la detección obtenida en otro (la voz). A partir de ese momento, la gran mayoría de trabajos que detectaba emociones en un solo canal reconocía como trabajo futuro el ampliar ese número de canales, debido precisamente a los beneficios que podía aportar la multimodalidad.

A pesar de esto, hoy en día la multimodalidad aún sigue en pañales y los investigadores que han intentado llevarla a cabo han terminado procesando cada canal de forma individual, sin realizar una correcta fusión de los datos [11]. Las causas de esto son de lo más variadas.

- *Lagunas en la teoría base*. En el propio campo de la Psicología hay conceptos que no han sido puestos a prueba. Por ejemplo, no hay evidencia de que, durante una interacción, una persona preste más atención a un canal que a otro; cada canal tiene peculiaridades que no siempre es posible capturar; no se ha explorado la posibilidad de que las personas atiendan a tipos de información más a que a canales de comunicación concretos, etc.
- *Tecnología inadecuada*. Si bien actualmente este problema ya está casi superado, ha sido un lastre importante en los años previos.
- *Demasiada información*. En cuestión de detectores de emociones, existen muchos servicios y aplicaciones disponibles, cada uno con sus propias peculiaridades: forma de funcionar, formato de los resultados, tiempo de respuesta, disponibilidad, etc. Esta cantidad de información se amplía si también consideramos hardware para detectar señales *fisiológicas* en el propio usuario. Esto fomenta que los investigadores empiecen a trabajar con un único canal de información y no lleguen a considerar combinarlo con otros.
- *Mayor complejidad*. La detección de emociones en varios canales no solo implica una mayor dificultad en lo que a preparación de equipo hardware se refiere, sino también a nivel de software. Los algoritmos que se preparen deben lidiar con varios tipos entradas y salidas, deben realizar una fusión de datos, deben tomarse decisiones dinámicamente en base a esos datos, etc. Esto se traduce en más tiempo, lo que a su vez significa más *coste*.

Con la realización de este TFM se pretende atacar esos dos últimos problemas, haciendo más accesible la detección multimodal para aquellos que quieran intentar utilizarla.

2.4 Combinación de clasificadores

Si observamos la propuesta realizada, concretamente la Figura 2, vemos que lo que se está definiendo se parece mucho a lo que en Minería de datos se denomina como *ensemble*, esto es, una combinación de clasificadores.

Los métodos de «ensamblado» de clasificadores son un conjunto de técnicas que se utilizan en el campo del aprendizaje automático para obtener clasificadores óptimos mediante la combinación de otros clasificadores [12]. Una de esas técnicas de ensamblado es el *bagging*. Tal y como se ilustra en la Figura 10, el *bagging* consiste en, partiendo de un mismo conjunto de datos, entrenar *distintos clasificadores con distintos subconjuntos* del conjunto de datos inicial, de manera que, a la hora de clasificar un nuevo dato, cada uno de esos clasificadores devuelva una predicción y todas estas se agreguen usando algún mecanismo de agregación.

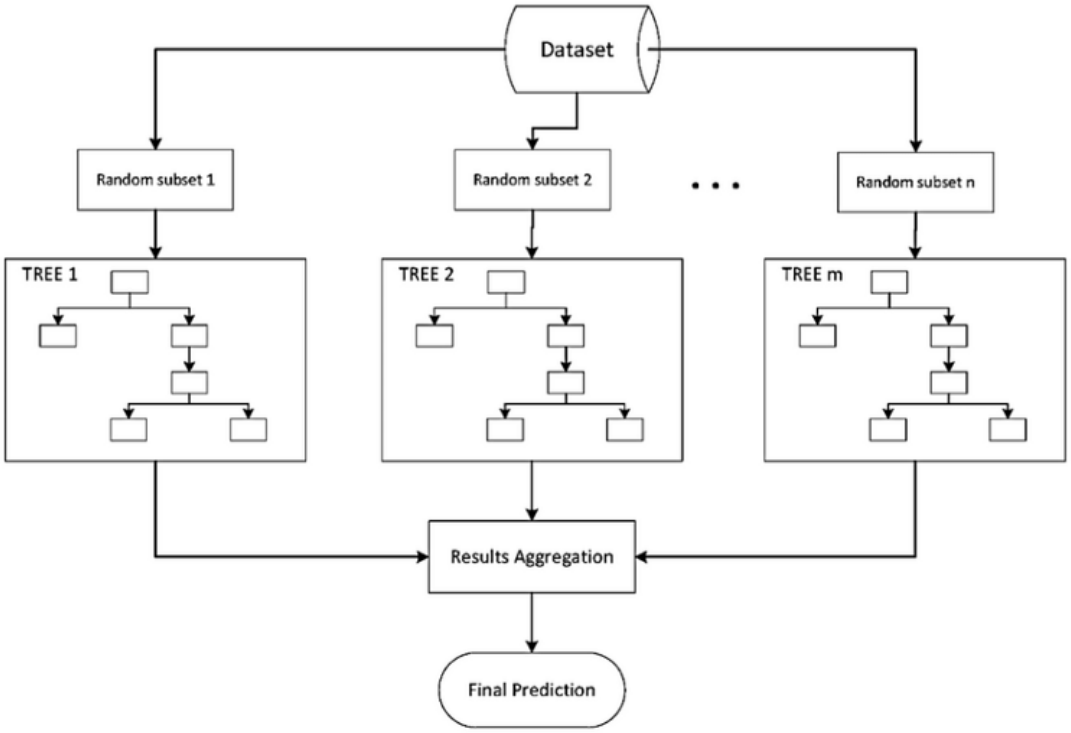

Figura 10. *Ensembler* de clasificadores [12]

Si recordamos una vez más la Figura 2, vemos que el sistema descrito en ella se parece mucho a la definición de *ensembler*, con la salvedad de que los detectores que utilizamos han sido entrenados por separado por datos que no provienen del mismo conjunto. Aun así, la fase posterior de predicción de un nuevo dato coincide exactamente

con lo que se plasmó en aquella figura. En la Figura 11 podemos ver una adaptación de la Figura 2 en la que se ha seguido el formato de la Figura 10.

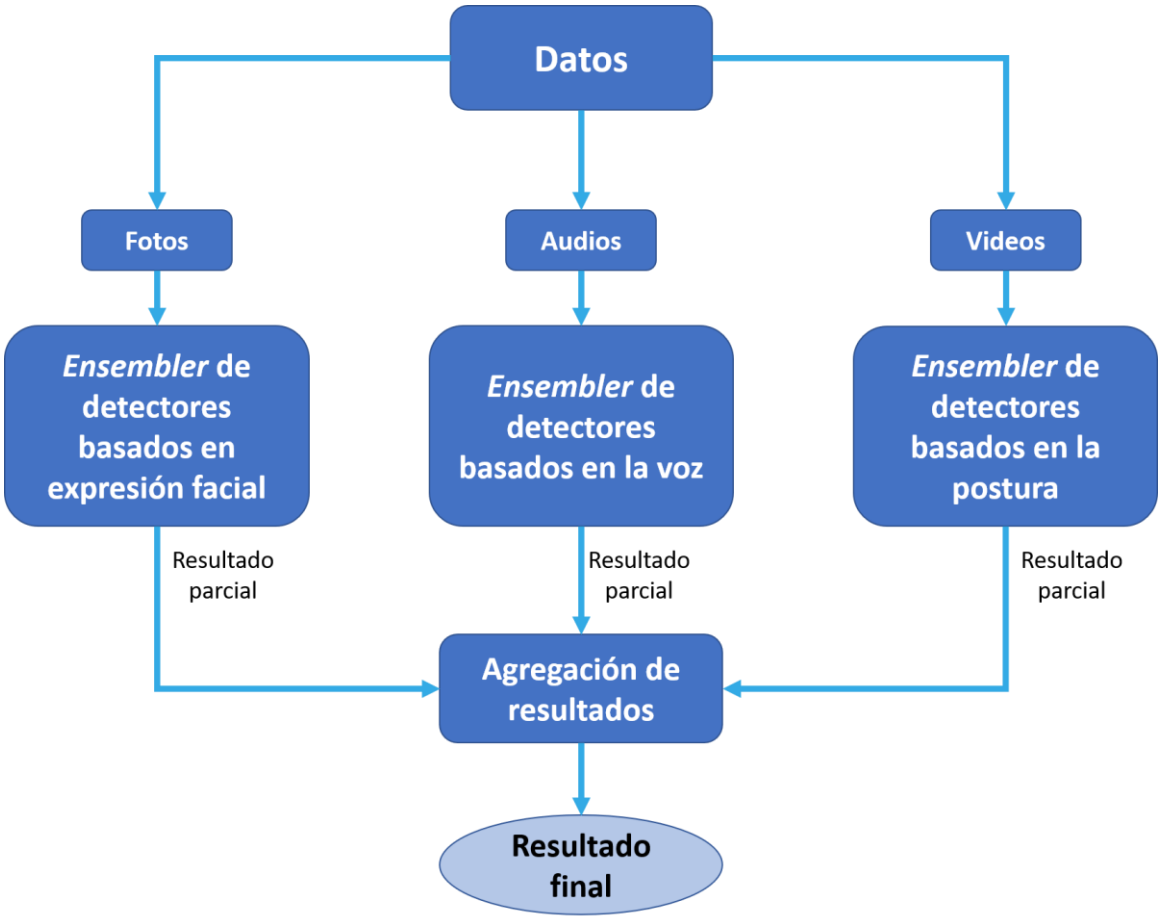

Figura 11. Propuesta adaptada

Cabe destacar el hecho de que el ensamblador de detectores que proponemos en este trabajo **no sigue totalmente la filosofía del *bagging***, puesto que en la fase de agregación no se hace tanto una votación para decidir qué respuesta es la correcta sino más bien una fusión de resultados al nivel que requiera el usuario del sistema (de datos, de decisión, etc. [3]).

La novedad que supone la propuesta de este trabajo es que cada **detector es a su vez otro *ensembler* de clasificadores**, considerando que los detectores combinados son del mismo tipo, esto es, todos detectan emociones en base al mismo canal afectivo.

2.5 Conclusiones

Como se ha adelantado en las subsecciones anteriores, la propuesta de este TFM consiste en un *marco de trabajo* que permita integrar de forma sencilla distintos detectores de emociones, para facilitar el acceso a tecnología de detección multimodal.

De forma experimental, se va a llevar a cabo esa detección multimodal de forma múltiple. En otras palabras, la agregación de datos se hará en dos niveles: primero a nivel de categoría (fusionamos los resultados de detectores del mismo canal, véase expresión facial, voz, texto, etc.) y por último, de forma general (fusión de esos resultados parciales).

En conclusión, el marco de trabajo desarrollado sigue el espíritu de un *ensembler* de clasificadores realizando un proceso de *bagging*, tanto a nivel de canal de información como a nivel global, con la salvedad de que no se realiza un voto por mayoría para cada clasificación.

CAPÍTULO 3. PROPUESTA DE SOLUCIÓN

En este capítulo se va a presentar el marco de trabajo que se ha desarrollado durante la realización de este TFM. Se revisará la metodología que guiará el desarrollo, a saber, Scrumban, las tecnologías que se utilizarán para crear dicho marco de trabajo y las herramientas de apoyo que se utilizan para gestionar el proyecto en sí.

3.1 Metodología

Para organizar el desarrollo del marco de trabajo propuesto fue necesario considerar una metodología de trabajo concreta. Por un lado, distinguimos entre metodologías tradicionales, basadas en planes, y metodologías ágiles, basadas en cambios y muy populares hoy día:

- **Metodologías dirigidas por planes (plan-driven)**. En una metodología basada en planes disponemos de toda la información necesaria antes de empezar el proyecto, por lo que se puede hacer una planificación a largo plazo. Son metodologías en las que se minimiza la incertidumbre a base de información. Son metodologías poco o nada flexibles, siendo esta característica lo que dio lugar al nacimiento de las metodologías ágiles. Aptas para proyectos grandes y de larga duración [13] [14].
- **Metodologías dirigidas por cambios (change-driven) o ágiles**. Este tipo de metodologías surgió como una respuesta a los problemas provocados por el tipo de metodología anterior (rigidez, demasiado tiempo invertido en documentar el proyecto, etc.). Las metodologías ágiles están hechas para responder a los cambios y a la incertidumbre, en contraposición a las anteriores, y buscan entregar valor al cliente en todo momento a través de entregas periódicas. La cantidad de incertidumbre se reduce según el proyecto se acerca a su fin. Más adecuadas para proyectos cortos y pequeños [13] [14].

Teniendo en cuenta esta información, el siguiente paso era analizar las características del proyecto:

- **Equipo pequeño**. El equipo de desarrollo está formado por una única persona, con los tutores del trabajo como *product owners*.
- **Proyecto corto**. La duración del proyecto es en principio de solo cinco meses.
- **Dependencia tecnológica**. Aunque el framework a desarrollar va a ser agnóstico respecto a los detectores que podrá integrar, se dependerá de detectores de emociones de terceros de cara al desarrollo de pruebas. Si estos no estuvieran disponibles, habría que desarrollar prototipos de detectores para las pruebas, lo que se traduciría en cambios y en más tiempo de desarrollo.
- **Enmarcado en un grupo de investigación**. El grupo de investigación en el que se enmarca este proyecto sigue una metodología de trabajo que incluye reuniones semanales de control.

Dadas estas circunstancias, una metodología ágil se adecúa mucho mejor al trabajo que hay que desarrollar. Dentro de las metodologías ágiles tenemos muchas opciones: eXtreme Programming (XP), Scrum, Lean, Crystal, Desarrollo Guiado por Pruebas (TDD), etc. Puesto que el TFG que precede a este TFM se desarrolló siguiendo Scrumban, una metodología ágil

que aúna Scrum y Kanban, va a optarse por elegir esta metodología una vez más. Los aspectos de esta metodología se detallarán en la siguiente subsección.

3.1.1 Scrumban

Scrum es una metodología que nació condensando los valores del Manifiesto Ágil [15]. Este manifiesto enfatiza los siguientes valores [16] [4]:

- **Individuos e interacciones sobre procesos y herramientas**. Un error que se comete en las metodologías más tradicionales es valorar sobremanera los procesos y las herramientas que se utilizan. Procesos y herramientas son elementos que han de apoyar el desarrollo: estos han de trabajar para los desarrolladores, no al revés. Es por ello que el primer principio da más valor a los individuos, que son los que trabajarán en satisfacer las necesidades de los clientes. De igual manera, si el desarrollo queda dirigido por los individuos, la comunicación entre los mismos será más fluida y natural, mientras que, si la comunicación está dictada por el proceso, esta será más artificial.
- **Software funcional sobre documentación extensa**. En metodologías tradicionales, la elaboración de documentación era un paso tedioso que tomaba una gran cantidad de tiempo. En las metodologías ágiles no se elimina la documentación, puesto que es un elemento útil y necesario, sino que se realiza de forma paralela al trabajo de desarrollo de manera que proporciona a los desarrolladores la información que necesitan sin detenerlos en minucias.
- **Colaboración con el cliente sobre negociación de contratos**. En enfoques más clásicos, el cliente acordaba con el director del proyecto cómo quería que fuese el producto final, especificando con gran detalle todos los aspectos de este producto, y sellándose estas características con un contrato. Todos estos acuerdos se realizaban antes de empezar el trabajo, de manera que era más fácil que se dieran casos en los que, incluso haciendo exactamente lo que el cliente pidió, no se satisficieran sus necesidades. El Manifiesto Ágil aboga por un cliente que participe en el proceso de desarrollo, ya sea involucrándose en el desarrollo de forma diaria o participando en pruebas y demos de forma regular.
- **Responder al cambio sobre seguir un plan**. En lugar de dedicar una gran cantidad de tiempo a elaborar planes extensos, detallados y llenos de interdependencias, se establecen periodos de trabajo breves en los cuales se van detallando los objetivos más prioritarios, que van cambiando según avanza el proyecto.

Scrum

El proceso de Scrum es un proceso *iterativo* basado en *sprints*. Un sprint es un periodo de tiempo corto, de entre dos y cuatro semanas, en el que se desarrolla un subconjunto de la funcionalidad solicitada por el cliente. En un sprint se producen incrementos, esto es, se hace un *añadido* al producto final, añadido completamente funcional y que aporta valor a ese producto. La funcionalidad que va a añadirse al producto durante un sprint está definida en el *sprint backlog*, que a su vez es un subconjunto del *product backlog*. El *product backlog* es la lista de requisitos que el cliente ha determinado que el producto final ha de cumplir. Esta lista, mantenida por el *product owner*, está en todo momento organizada y priorizada y ha de estar visible y accesible para los implicados en el proyecto. Antes de empezar el sprint, el product owner (que puede ser el propio cliente o un representante de este) se reúne con el equipo de desarrollo para indicar qué requisitos son más prioritarios y cuáles deberían acometerse cuanto antes, y se llega a un acuerdo sobre lo que se va a implementar en el siguiente sprint. Este subconjunto de requisitos que van a implementarse en el sprint conforma el *sprint backlog*. El grupo de desarrollo transformará a su vez ese subconjunto de requisitos en *tareas*, que serán asignadas a los miembros del equipo para su desarrollo. En la Figura 12 podemos ver una representación gráfica de todo este proceso.

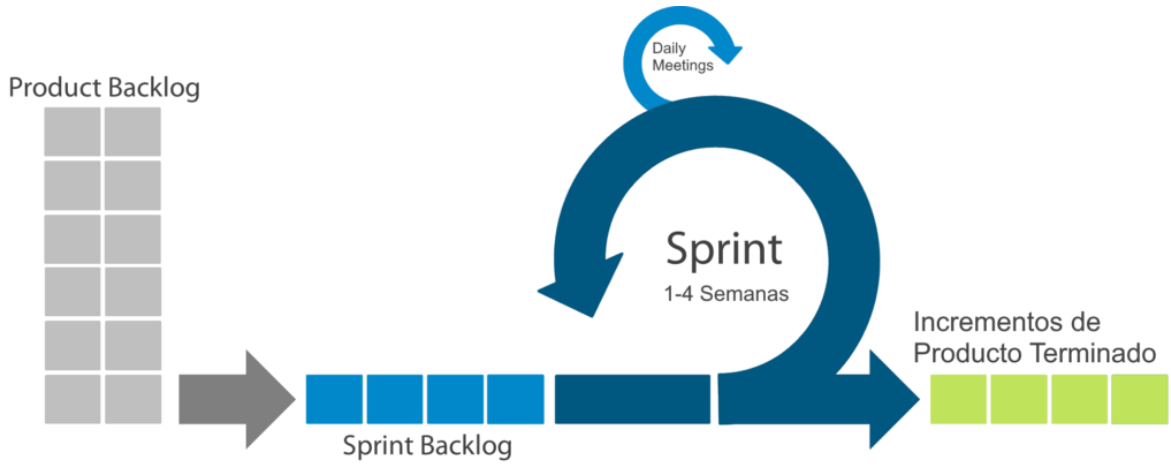

Figura 12. Proceso iterativo Scrum

Además de las dos reuniones previas al comienzo del sprint, detalladas en el párrafo anterior, Scrum define otros tres tipos de reuniones. En primer lugar, tenemos las r*euniones diarias*, que son reuniones muy breves (no más de 15 minutos) que se mantienen cada día que dura el sprint para que el equipo pueda poner en común su progreso, saber si alguien necesita apoyo, etc. En segundo lugar, tenemos la *reunión con el product owner* al acabar el sprint, en la que cual se le enseña la funcionalidad que se ha añadido al producto. Por último, entre esa reunión y la siguiente reunión de planificación del sprint, tenemos la reunión con el *scrum master*. El scrum master es una persona que se encarga de que Scrum se siga correctamente durante todo el tiempo que dure el proyecto. Esta reunión la mantiene el scrum

master con el equipo para revisar como ha sido su desempeño en el sprint desde el punto de vista de Scrum.

Kanban

El sistema Kanban como tal es un sistema de visualización de información aplicable a cualquier proceso de producción. Este sistema se basa en utilizar *tarjetas* o *carteles* que se pegan a los bienes producidos para indicar en que punto del proceso de producción están, siendo esta información visible en todo momento en el *tablero Kanban*. Esta metodología, nacida en la empresa Toyota en los años cincuenta, fue adaptada al mundo del desarrollo del software hace poco tiempo [17].

Si bien la metodología de desarrollo Kanban apenas impone artefactos, roles o reuniones, sus valores principales son muy útiles y estimulan la realización del trabajo, lo que explica su popularidad y su integración con otras metodologías. Así, las características principales de Kanban son las que siguen [18]:

- **Visualización del trabajo**. Mediante el uso de un *tablero Kanban* físico o virtual todos los trabajadores están al corriente del trabajo que hay pendiente, de lo que tienen que hacer, del progreso del proyecto, etc. En el tablero Kanban se distinguen tantas columnas como fases se tengan en el proceso de producción (pendiente, en proceso, en pruebas, en despliegue, etc.) y las tareas de cada fase se representan mediante tarjetas, usándose colores o símbolos para indicar su prioridad, su riesgo, su importancia, etc. En la Figura 13 podemos ver un ejemplo de este tablero.
- **Limitación del trabajo en progreso**. Kanban exige que se establezca un máximo de tareas que puede haber activas en cada momento. Este límite, denominado WIP (*Work In Progress*) se establece para cada fase del proceso de producción de forma realista, de tal forma que las tareas más lentas no acaben bloqueando el paso de otras tareas más livianas.
- **Entrega continua**. A diferencia de otras metodologías que establecen plazos y periodos de tiempo para realizar el trabajo acordado y hacer las entregas establecidas, Kanban solo se centra en la realización del trabajo.

El poder de apoyo de Kanban reside en el componente visual del trabajo: al representar todo el flujo de trabajo en un tablero accesible para todos, usando códigos de colores para identificar características de las tareas, se consigue un incremento de la productividad, se estimula la colaboración y la autoasignación de tareas, etc.

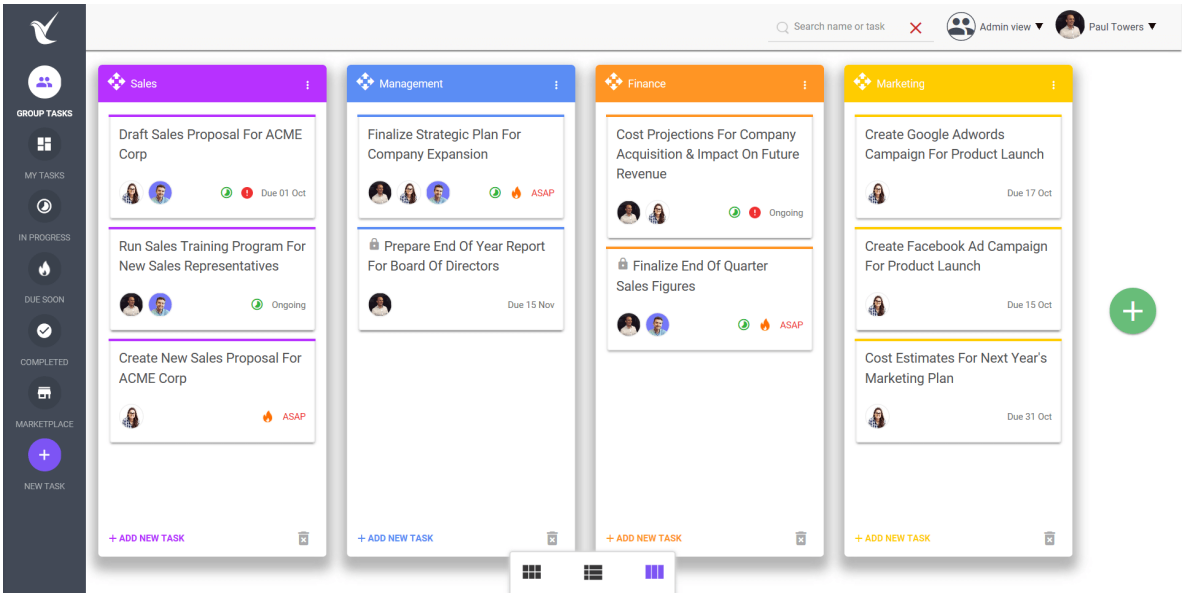

Figura 13. Tablero Kanban

Dadas estas características, Kanban supone un apoyo perfecto para cualquier metodología ágil, por lo que desde su popularización se ha usado sobre todo en combinación con otras. Una de esas fusiones es **Scrumban**, que usa Scrum como base, pero se aprovecha de ese componente visual y esa limitación del trabajo que impone Kanban. Herramientas como *Kunagi* [19] ya vienen dando soporte a esa fusión desde antes que existiese, puesto que Kunagi ofrece un panel para ver las tareas de un sprint agrupadas en columnas (pendientes, en progreso, hechas), como se puede ver en la Figura 14.

Figura 14. Tablero Kanban en Kunagi

Si bien Kunagi no da soporte a la limitación de tareas, es un buen punto de partida como herramienta. Otra opción popular, y de hecho una de las que va a utilizarse, es *Trello* [20], que da soporte para la creación de tableros, tarjetas y listas de tarjetas, como veremos a continuación.

3.2 Tecnologías utilizadas

En los siguientes apartados se presentan las tecnologías que se han utilizado durante el proyecto, tanto las herramientas usadas para gestionar el proyecto en sí como las herramientas utilizadas durante la implementación.

3.2.1 Entorno Node.js

Node.js es un entorno de ejecución de JavaScript orientado a *eventos asíncronos*, pensado precisamente para construir aplicaciones web escalables [21]. Así, en lugar de usar un sistema basado en hilos del sistema operativo que sea, que puede llegar a ser muy ineficiente, se utiliza un sistema basado en *callbacks*: se define la función que ha de llamarse cuando llega una determinada petición HTTP y cuando esta llega, se ejecuta; mientras no lleguen peticiones, el proceso de Node.js está dormido. La versión de Node.js que va a utilizarse es la última versión estable, a saber, la versión 10.15.0.

Incluso si no llegase a aprovecharse ese poder de concurrencia, Node.js y JavaScript fueron las tecnologías base usadas para el desarrollo del TFG del autor [4], por lo que se cuenta una mayor experiencia con ellas y se posee un conocimiento más sólido de la lógica basada en eventos asíncronos y *callbacks*. Dentro del entorno de Node.js se utilizará el *framework* Express [22] para el desarrollo del *framework* propuesto.

3.2.2 Face++

Si bien el sistema propuesto (marco de trabajo o framework indistintamente de ahora en adelante) será tecnológicamente agnóstico, se necesitan unos cuantos servicios de cara a fases de prueba y desarrollo. Para la *expresión facial* se utilizará el servicio de detección de emociones ofrecido por la empresa **Face++** [23], que ofrece su servicio a través de una API.

La API de Face++ es capaz de procesar imágenes que recibe en la propia petición HTTP, devolviendo como respuesta un objeto JSON en el que expresa, con rangos que van de 0 a 100, el grado de *alegría, neutralidad, disgusto, tristeza, sorpresa, enfado* y *miedo* que ha detectado.

3.2.3 Beyond Verbal

Para utilizar al menos dos tipos de detección, se ha recurrido también a detección de emociones basada en la voz. Para este propósito se ha utilizado el servicio ofrecido por la empresa **Beyond Verbal** [24], cuyos servicios ya se usaron durante el TFG [4].

Al igual que Face++, Beyond Verbal ofrece sus servicios en forma de API. A través de una petición HTTP, se envía el audio que se quiere analizar a los servidores de Beyond Verbal. Antes de analizar el contenido de ese archivo de audio, la API aplica un leve procesamiento para optimizar dicho archivo. Este procesamiento implica recortar aquellas zonas en las que no se detecta una voz hablando, de manera que todo el análisis de emociones posterior se centre solo en la voz que se va a analizar (Figura 15).

En cuestión de resultados, Beyond Verbal nos devuelve cuatro métricas, a saber, *Humor*, *Valencia*, *Excitación* y *Grupo Emocional*. Las tres primeras se expresan dualmente mediante un valor *cuantitativo* en el rango [0,100] y un valor *cualitativo* que puede ser «Bajo», «Medio» o «Alto». Por otra parte, la métrica *Grupo Emocional* indica, usando el lenguaje natural, que emoción o emociones se han detectado, como una versión más fácil de interpretar de los datos ofrecidos por las tres primeras métricas [4].

- **Humor** (*Temper*). Esta métrica indica el temperamento que se ha detectado en la voz. Los valores altos de está métrica están asociados a emociones *agresivas*, mientras que los valores bajos están asociados a emociones *depresivas*.
- **Valencia** (*Valence*). Esta métrica nos indica el grado en el que una emoción es positiva o negativa. Una valencia alta representa emociones *positivas*, como la alegría, mientras que los valores bajos expresan emociones *negativas*, como la tristeza.
- **Excitación** (*Arousal*). Esta métrica expresa la energía que se ha detectado en la voz analizada. Los valores bajos están asociados a tonos de voz que expresen aburrimiento o cansancio, mientras que los valores altos reflejan excitación (pasión comunicativa, nerviosismo, ira).
- **Grupo emocional** (*Emotion Group*). Este valor ofrece una etiqueta en inglés que indica a qué grupo de emociones pertenece la emoción que se ha detectado. Un ejemplo del tipo de valor que toma esa etiqueta podría ser “Tristeza/Aburrimiento/Sin energía”.

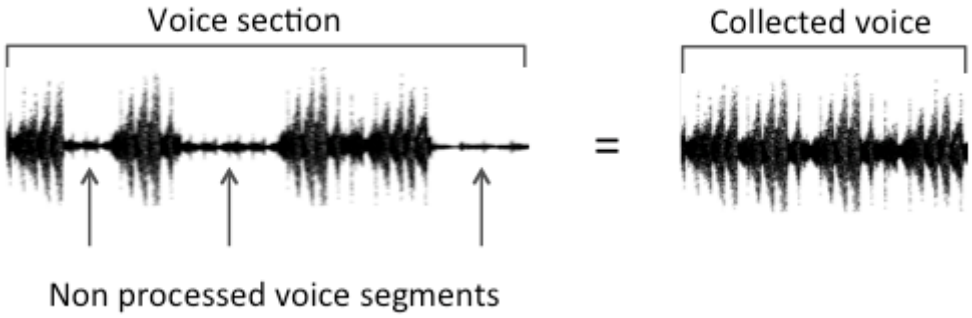

Figura 15. Procesamiento de audio en Beyond Verbal

3.2.4 Postman

El framework que se ha desarrollado se ofrece en forma de API. Una de las razones que motivó esta decisión fue el hecho de que una API abre la puerta al desarrollo de aplicaciones en cualquier plataforma y con cualquier tecnología.

Para el desarrollo y prueba de la API se ha utilizado la herramienta **Postman** (Figura 16). Postman [25] ofrece soporte para realizar peticiones HTTP mediante una interfaz que permite modificar todos los parámetros de la petición (cabeceras, argumentos de la petición, etc.). Postman nos ahorrará tiempo y esfuerzo a la hora de probar los *endpoints* de la API, pues de lo contrario habríamos de usar herramientas como *curl*, mucho menos usable, o tendríamos que hacer nuestros propios scripts para cada petición.

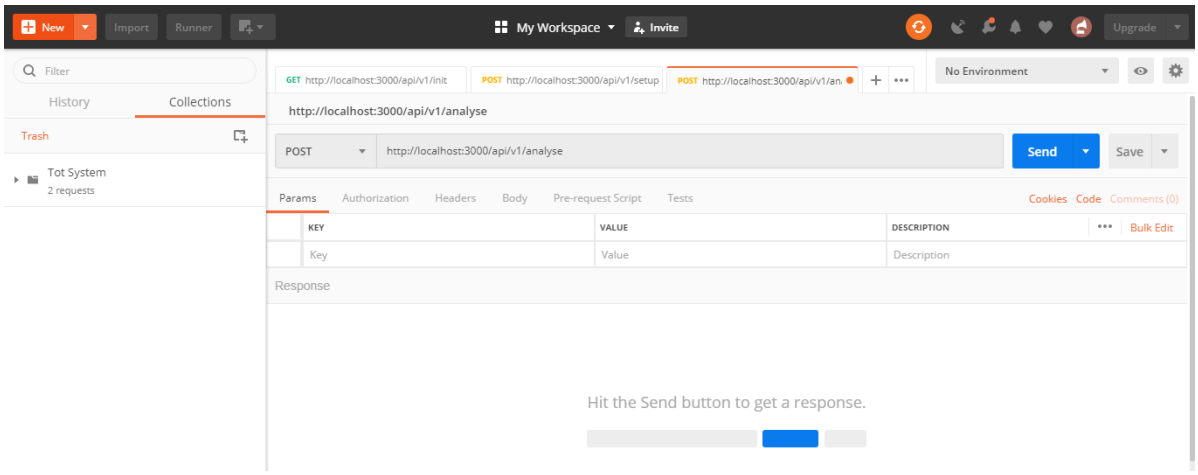

Figura 16. Interfaz Postman

3.2.5 Visual Studio Code

El Entorno de Desarrollo Integrado (Integrated Development Environment, IDE) que vamos a utilizar será **Visual Studio Code** [26]. Este editor, gratuito y de código abierto, resulta muy potente y ligero. Gracias a su sistema de *extensiones*, Visual Studio Code puede personalizarse y convertirse en una poderosísima herramienta para el desarrollo de cualquier tipo de aplicación. En nuestro caso, se han utilizado extensiones de apoyo para mantener el código con el formato adecuado (ESLint, Beautify), para gestionar el repositorio en el que se sube el código (GitLens), etc.

3.2.6 Herramientas de soporte al desarrollo del proyecto

Durante el desarrollo del framework indicado se han utilizado otras herramientas además de las anteriormente mencionadas. Estas herramientas son las que siguen:

- **JSDoc**. JSDoc es una API para generar documentación de JavaScript. Haciendo uso de distintas etiquetas como *@function*, *@param* o *@return* al documentar el código escrito, JSDoc es capaz de compilar dicha documentación en un conjunto de ficheros HTML, CSS y JavaScript. Así, con una configuración nula, podemos generar la documentación de una aplicación en forma de archivos de una página web.
- **GitHub**. Todas las tareas de programación se han llevado a cabo con GitHub y Git como soporte. GitHub, gracias a su gestor de *Issues*, permite llevar un seguimiento de los fallos, problemas y/o peticiones de funcionalidad que hayan surgido. Entre otras cosas, también permite crear una página colaborativa con información asociada al proyecto. Estas y otras funcionalidades lo convierten en una herramienta imprescindible en las fases de desarrollo. Además, se ha utilizado la tecnología *Github Pages* que ofrece el propio GitHub para hospedar archivos y mostrarlos como si de una página web se tratara. En nuestro caso, se ha utilizado esta tecnología para hospedar la documentación que se ha generado con JSDoc [27].
- **Trello**. Trello es una herramienta enfocada en la gestión de proyectos que nos permite crear tableros, listas y tarjetas. Las tarjetas en Trello no son como un simple *pósit* de papel, sino que pueden etiquetarse con distintos colores, asignarse a personas, adjuntarse archivos en ellas, etc. En nuestro caso, utilizaremos Trello para crear un tablero Kanban, en el que distinguiremos tres columnas, a saber, «Tareas Pendientes», «En Proceso» y «Tareas Hechas».

- **Microsoft Project**. De cara a controlar el avance del proyecto y de los distintos sprints, se utilizará Microsoft Project 2019 para controlar y actualizar el cronograma preparado durante el anteproyecto.

3.3 Consideraciones legales y tecnológicas

Tal y como dicta el Real Decreto 1720/2007, de 21 de diciembre, por el que se aprueba el Reglamento de desarrollo de la Ley Orgánica 15/1999, un dato de carácter personal es "*cualquier información numérica, alfabética, gráfica, fotográfica, acústica o de cualquier otro tipo concerniente a personas físicas identificadas o identificables*" [28]. Por tanto, a la hora de manipular esta información hay que tener en cuenta las imposiciones que haga la ley en lo que respecta al *tratamiento* que hagamos.

Si bien los detectores de emociones rara vez almacenan los datos que reciben, el mero hecho de enviar una imagen o un archivo de audio a un servidor de terceros ya supone un tratamiento, lo que nos obliga, como desarrolladores del framework, a cumplir las imposiciones que hace la Ley Orgánica de Protección de Datos y, en el caso de España, la Agencia Española de Protección de Datos (AEPD) en materia de seguridad.

Debido a esto, desarrollar la aplicación como un framework centralizado que respetase todas las imposiciones legales podría añadir severas dificultades a la implementación, por lo que esta primera versión se planteará como un sistema que cada usuario desarrollador despliega en sus propios servidores y con sus propias condiciones. Queda así para una versión futura la implementación de sesiones para distintos usuarios, las conexiones seguras entre usuarios y servidores, almacenamiento de credenciales, tratamiento seguro de los archivos que contengan información de carácter personal, registro de ficheros en la AEPD, etc.

También merece la pena comentar que, aunque la aplicación sea tecnológicamente agnóstica, se dependerá de servicios de terceros para poder hacer pruebas sobre la funcionalidad de esta, lo cual podría limitar las actividades de desarrollo. Por ejemplo, las dos herramientas de detección de emociones utilizadas se han usado durante periodos de evaluación, pues no ofrecen licencias gratuitas. Asimismo, Beyond Verbal es aún más rígida que Face++, aceptando solo archivos de audio que cumplan ciertas condiciones. Para no desviar el desarrollo hacia el procesamiento de audio con Node.js, se han utilizado algunos de los ficheros de prueba que ofrece la propia compañía.

3.4 Conclusiones

En este capítulo se ha presentado la metodología por la que se va a regir el desarrollo del prototipo, a saber, Scrumban, una combinación de Scrum y Kanban.

Se han revisado también las tecnologías que van a utilizarse, distinguiendo cuáles van a usarse para gestionar el proyecto, cuáles van a usarse para hacer pruebas de la aplicación en términos de funcionalidad y cuáles van a usarse para el desarrollo.

Por último, se han comentado algunas de las limitaciones legales y/o económicas de este proyecto, justificando así el porqué de ciertas decisiones clave del desarrollo del sistema.

CAPÍTULO 4. DESARROLLO DEL SISTEMA

En este capítulo se detallará el proceso a través del cual se ha desarrollado la aplicación del TFM. El proceso de desarrollo, realizado bajo el marco de trabajo de Scrumban, consta de tres sprints. A lo largo de las siguientes páginas veremos cómo se afrontó el desarrollo, así como las decisiones que lo dirigieron.

Para facilitar la lectura de este capítulo, la documentación y diagramas de UML se han colocado en el ANEXO A.

4.1 Introducción al desarrollo

En este primer apartado se presentan las adaptaciones que se han realizado sobre la metodología de trabajo empleada, así como los stakeholders[1] del proyecto y la estructura de los siguientes apartados, donde se detalla la evolución de este.

A continuación, se describen las adaptaciones realizadas a Scrumban para este proyecto.

Adaptaciones de las metodologías

Dado que el presente proyecto ha sido desarrollado por un equipo de una única persona durante el transcurso de cinco meses, se han realizado algunas adaptaciones sobre las metodologías aplicadas.

- **Se prescinde de las reuniones diarias de Scrum**. Puesto que el equipo consta de una única persona, no es necesario hacer estas reuniones.
- **Se prescinde del rol de Scrum Master**. Puesto que los conocimientos sobre Scrum y su aplicación se consolidaron durante el TFG, en este trabajo se prescindirá de este rol, así como de la reunión que lo involucra.
- **Se produce un solapamiento de los roles impuestos por Scrum.** El rol de equipo de desarrollo será desempeñado por el autor de este trabajo, mientras que el rol de product owner será desempeñado conjuntamente por el autor y los directores de este trabajo.
- **La limitación de trabajo en progreso.** Se decide aplicar el Trabajo en Progreso (WIP para futuras referencias) solo a las tareas que estén en la fase de desarrollo. La cantidad de tareas que puede haber en desarrollo como máximo será la mitad de tareas que hubiera inicialmente en el sprint backlog (SB para futuras referencias). Si hay tareas que se pueden clasificar en distintas categorías, se distinguirá una sublimitación dentro de esa categoría, estando determinada esa sublimitación por la proporción de tareas que suponga esa categoría sobre el total de tareas. Esta cantidad queda determinada por la ecuación de vemos a continuación.

$$WIP_{categoria\ i} = \frac{NT_{categoria\ i}}{NT_{SB}} * WIP_{global}$$

[1] Persona interesada en el proyecto.

Siendo NT_{SB} el número de tareas que tiene inicialmente el SB y $NT_{categoría\ i}$ el número de tares de esa categoría que había inicialmente en el SB.

Si solo hay una categoría, el WIP de esa categoría coincidirá con el WIP global, estando este determinado por la ecuación presentada a continuación. Si, por ejemplo, un sprint está compuesto únicamente por tareas de implementación y desarrollo, y hay 10 tareas en el SB, entonces el WIP global será de 5. Si además de esas 10 tareas de implementación, tenemos 8 de documentación, entonces el WIP de las tareas de implementación será de 5 y el de las tareas de documentación será de 4.

$$WIP_{global} = \frac{NT_{SB}}{2}$$

A la hora de determinar el WIP, no se ha tenido en cuenta la duración de las tareas, debido a que los tiempos de duración son muy similares.

Stakeholders identificados

Al decidirse la temática y la dirección que iba a tomar el proyecto, se identificaron los principales stakeholders del proyecto.

- **Cliente**. El rol del cliente es desempeñado por los directores del proyecto. Su colaboración en el proyecto es vital, ya que en colaboración con ellos se establecían requisitos, se acordaba el rumbo que debía seguir el proyecto, etc.
- **Desarrolladores.** A diferencia de la aplicación desarrollada durante el TFG [4], que fue una aplicación final, la propuesta en este TFM consiste en un framework, esto es, una herramienta que otros puedan usar para hacer aplicaciones finales. Por este motivo, los posibles futuros desarrolladores de esas aplicaciones finales forman parte del conjunto de personas interesadas o afectadas por el proyecto.
- **Investigadores**. Como se identificó en el CAPÍTULO 2, en materia de investigación todos los trabajos reconocen la superioridad de los detectores de emoción multimodales frente a los unimodales. No obstante, estos detectores entrañan también una mayor complejidad, motivo por el cual su uso no está tan extendido como cabría pensar dada su superioridad. Puesto que este framework busca solventar parte de su problemática, hay que considerar los intereses de aquellos investigadores que quisieran usar el framework para introducir la detección multimodal de emociones en sus investigaciones.
- **Empresas de servicios de detectores de emociones**. Si la detección multimodal de emociones se popularizara, las empresas que ofrecen estas tecnologías podrían estar interesadas en colaborar juntas para facilitar aún más

la integración de sus servicios: podrían proponerse integraciones, estándares en el tipo de resultados ofrecidos, paquetes de licencias más económicos para usar varios detectores juntos, etc. Esto las convierte en un ente más a considerar en el desarrollo.

Documentación del proyecto

Cada una de las siguientes subsecciones contendrá la información de cada *sprint*. Para la organización de la información se ha optado por la siguiente estructura.

Planificación

En este apartado se detallan el objetivo del sprint y su duración y se comentan decisiones previas que se tomasen relativas a lo que debía producirse al final del sprint. Las dos reuniones previas al sprint que impone Scrum se fusionan en una sola, dado el tamaño del equipo de desarrollo. En esta fase se hará especial hincapié en el sprint backlog, esto es, el conjunto de historias que se desarrollarán durante el sprint. Dicha historias se presentarán con el formato « Como <quién> Quiero <qué> Para <objetivo> [29]», acompañadas de una estimación de la dificultad de su implementación usando los *Puntos de Historia* como unidad [30].

Desarrollo

En este apartado se expone el trabajo realizado durante el sprint correspondiente, así como los artefactos que hayan podido generarse y los impedimentos que hayan podido surgir.

Revisión

Al final del sprint se realiza la revisión del incremento con el PO, quedándose recogido en este apartado una pequeña acta de dicha reunión.

4.2 Sprint 1. Revisión sistemática de la literatura

Durante este sprint se llevó a cabo una revisión sistemática de la literatura que permitió definir claramente las lagunas que se acusan en el campo de la aplicación de la Computación Afectiva. Estas lagunas permitieron tomar decisiones sobre el futuro de este trabajo con conocimiento de causa.

4.2.1 Planificación

Objetivo. Realizar una revisión sistemática de la literatura en términos de aplicación de la Computación Afectiva para fundamentar el trabajo y establecer la dirección del proyecto.

Duración. 21 días (24/09/2018 – 22/10/2018)

Aunque este sprint se ha imputado como si hubiera empezado el 24 de septiembre de 2018, el trabajo de exploración de la revisión sistemática comenzó algún tiempo atrás, puesto que en principio era un trabajo que se había pensado como parte de una publicación ajena a este TFM. No obstante, la carga de trabajo que exigía la revisión sistemática requería un tiempo de desarrollo que no podía satisfacerse plenamente en el tiempo de desarrollo del TFM. Por este motivo, se decidió reconducir esa revisión para adaptarla al contexto del TFM. Es también por esto por lo que los códigos de las historias del sprint backlog son «SA» y «SB» en lugar de usar claves numéricas y por lo que no se ha hecho una estimación en puntos de historia de la dificultad de estas (Tabla 2).

Tabla 2. Sprint backlog - Sprint 1

ID	Nombre	Descripción
SA	Revisión sistemática de la literatura	Como cliente quiero saber con cierto grado de certeza que trabajos se han realizado en el campo de la Computación Afectiva en términos de aplicación, de cara a encauzar el producto final.
SB	Análisis de los datos recogidos	Como cliente quiero calcular una serie de valores indicadores que me permitan comprender el «paisaje» global de la investigación realizada.

4.2.2 Desarrollo

Para afrontar la revisión sistemática se analizaron algunos ejemplos de revisiones ya publicadas, como [31] y [32], y manuales sobre cómo hacerlas [6].

Se investigó la posible existencia de revisiones sistemáticas cómo la que se pretendía hacer y, al ver que no se había hecho, se establecieron las preguntas de investigación, la estrategia de búsqueda y los criterios de selección.

Si bien no formaba parte de la metodología sobre la construcción de revisiones sistemáticas, se optó por registrar también los datos de los artículos rechazados, todo ello junto a una breve explicación sobre por qué se había rechazado cada artículo. Esto nos ayudaría a encontrar lagunas y deficiencias en el panorama investigador con mayor facilidad.

Una vez fue recogida toda la información en un fichero XLSX, se utilizó **Python 3.6**, junto con las librerías Matplotlib [33], Pandas [34] y Numpy [35], para representar gráficamente los datos hallados, para agregarlos y depurarlos, etc.

4.2.3 Revisión

Fruto del trabajo llevado a cabo durante este sprint tenemos el Capítulo 2 de este mismo documento. La revisión sistemática nos permitió identificar serias lagunas en todo lo que concierne a los sistemas multimodales y a su integración en proyectos reales, lagunas que modelaron el futuro del proyecto en los próximos meses.

Fue en este momento en el que se decidió el tipo de sistema que se iba a desarrollar. Como ya se ha adelantado varias veces, se decidió desarrollar un marco de trabajo o framework que permitiese a desarrolladores e investigadores del campo de la Computación Afectiva integrar detección multimodal en sus proyectos de forma más sencilla. Además de esta aportación, se estableció que la multimodalidad de este framework estaría dividida en dos faces. Por un lado, se agruparían los detectores del mismo tipo (detectores basados en expresión facial, en la voz, en la postura, etc.) y se fusionarían los resultados de cada grupo, para luego, finalmente, agregar esos resultados en una única respuesta.

Al finalizar esta revisión se acordaron ciertos aspectos técnicos sobre el marco de trabajo. En primer lugar, dicho marco sería implementado en forma de Interfaz de Programación de Aplicaciones (**API** por sus siglas en inglés, Application Programming Interface). Se eligió esta forma de implementación sobre otras, como sería una implementación en forma de Kit de Desarrollo de Software (SDK por sus siglas en inglés, Sofware Development Kit), por la flexibilidad y libertad que otorga a los desarrolladores el poder acceder a un servicio a través de peticiones HTTP estándares, en lugar de tener que buscar librerías o SDKs que sean específicas de la plataforma en la que estén programando. También se decidió que la tecnología usada sería **Node.js**, debido a la experiencia del autor con este entorno de desarrollo.

Durante la planificación inicial se acordó que este primer sprint duraría solo once días. No obstante, se decidió fusionarlo con el siguiente sprint planificado, de diez días de duración, puesto que la carga de este primer sprint resultó ser mayor de la esperada y la carga del siguiente sprint, por el contrario, parecía inferior.

Dado que el sistema que se planteó construir podría crecer mucho y muy rápido, se acordó realizar un análisis activo de sistemas modules y extensibles, para poder dotar a nuestro sistema de dichas características.

4.3 Sprint 2. Modelado del sistema multimodal y prototipo inicial

En el marco de este segundo sprint se analizaron distintos tipos de arquitectura y métodos de desarrollo, en busca de una arquitectura que se ajustase a las características principales del sistema propuesto, a saber, modularidad y extensibilidad.

Igualmente se abordó el desarrollo de un primer prototipo básico, que a su vez seguirá enriqueciéndose en futuros sprints.

4.3.1 Planificación

Objetivo. Investigar sistemas similares al propuesto e iniciar su modelado y desarrollo prototípico.

Duración: 25 días (23/10/2018 – 26/11/2018)

Tanto en la reunión de revisión del sprint anterior como en la reunión de planificación de este se recalcó el problema que iba a suponer la *no estandarización* de las tecnologías de detección. En otras palabras, el hecho de cada tecnología detecte las emociones en un fichero multimedia usando sus propias reglas y devolviendo unas métricas en su propio formato dificulta la integración de tecnologías. De las historias que conforman el sprint backlog de este sprint (véase Tabla 3), la historia S2 es la que resulta más crucial de cara al desarrollo del sistema.

El resto de las historias de este sprint están más enfocadas a desafíos técnicos, relacionados sobre todo con la forma de modelar la API a desarrollar.

En la Tabla 4 podemos ver la división de tareas que se hizo de las historias que formaban parte del Sprint Backlog.

Tabla 3. Sprint backlog - Sprint 2

ID	Nombre	Descripción	SP
S1	Revisión de sistemas modulares y extensibles.	Como cliente quiero saber qué otros sistemas y/o arquitecturas extensibles existen.	13
S2	Diseño de formato estándar.	Como cliente quiero establecer un formato o esquema común para que todas las tecnologías integradas se comuniquen de la misma forma.	5
S3	Modelado del sistema completo.	Como cliente quiero tener un sistema que sea capaz de integrar distintos canales de información de forma separada para luego integrar el resultado parcial de cada uno de ellos.	21
S4	Desarrollo del prototipo inicial.	Como cliente quiero disponer de una aplicación que me permita usar el sistema mencionado a nivel de usuario para crear mis propias aplicaciones con una sencilla integración de detectores de emociones.	21
		Total	60

Tabla 4. Tareas de Sprint backlog - Sprint 2

Nombre	Tarea	Horas	+/-
Revisión de sistemas modulares y extensibles.	Estudio de patrones de diseño aplicables	3	-1
	Análisis de sistemas existentes	9	0
Diseño de formato estándar.	Análisis de clasificación PAD	5	0
	Análisis de XML/JSON	2	0
Modelado del sistema completo	Modelado de detectores individuales	1	0
	Modelado de integradores de detectores	2	0
	Modelado de sistema al completo	3	+2
Desarrollo de prototipo inicial	Creación de proyecto base	3	+3
	Revisión de tecnologías disponibles	3	+1
	Desarrollo de prototipo inicial	4	+5
	Total	35	+10

4.3.2 Desarrollo

Como parte de la tarea sobre el análisis de sistemas existentes se encontró el sistema OpenRec [36]. OpenRec ofrece una librería modular de código abierto para estudiar la integración de algoritmos de recomendación. En la Figura 17 podemos ver un diagrama con la estructura de dicho framework. Si descartamos el bloque Utility, pues este está formado por bloques auxiliares, vemos que los bloques **Recommender** y **Module** se parecen mucho a lo que se deberían desarrollar en el framework propuesto.

Por un lado, tenemos el bloque **Recommender**, que es el bloque que provee de datos. Puesto que la propuesta de este trabajo consiste un marco de trabajo para la detección de emociones y el análisis de resultados, de este bloque se encargaría el desarrollador o investigador que estuviese usando la API desarrollada. Así, dicha API habría de implementar la funcionalidad asociada, en este caso, al bloque Module.

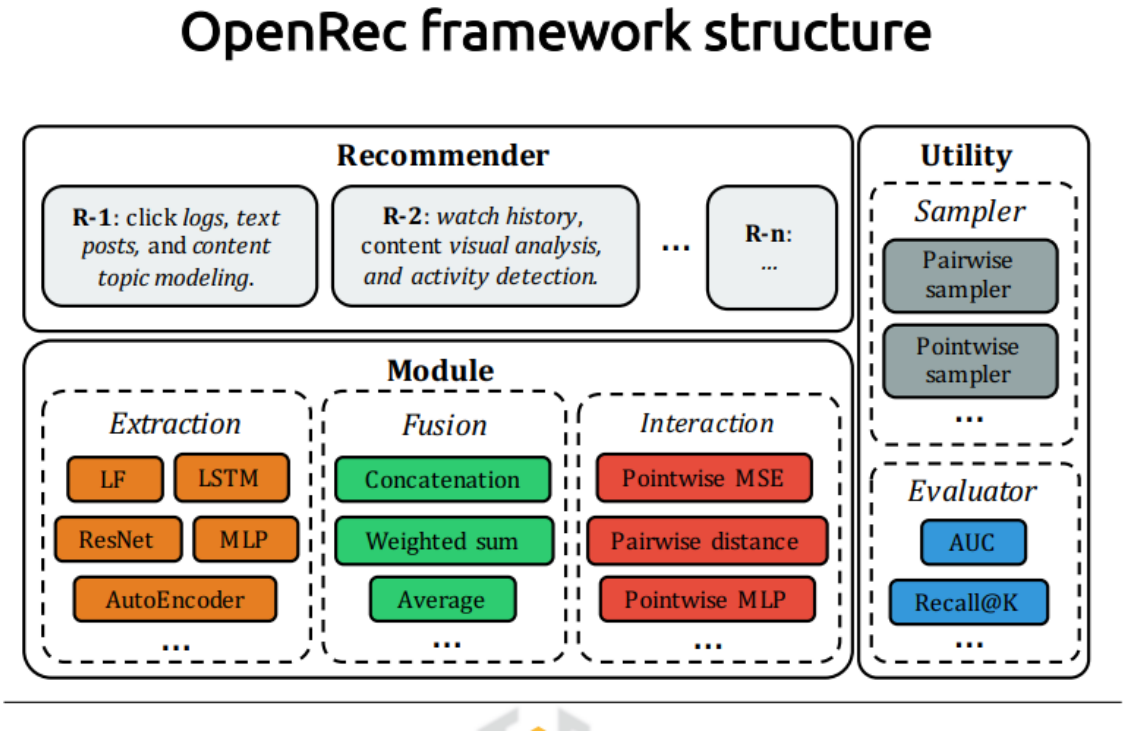

Figura 17. Estructura del framework OpenRec [36]

De hecho, si analizamos el esquema de funcionamiento de OpenRec de la Figura 18 vemos que se parece mucho a la idea inicial que desarrollamos en la Figura 2. En el nivel más inferior tenemos los datos, representados por *cuadrados grises*. Estos datos se corresponden con el bloque **Recommender** de la Figura 17. En el siguiente nivel tenemos distintos detectores y clasificadores, identificados como *círculos naranjas*, que nos devuelven predicciones y clasificaciones en base a los datos del nivel anterior. Estos detectores se corresponden con el bloque **Extraction** de la Figura 17. A continuación,

tenemos la fusión de resultados, representada mediante *pentágonos verdes*, que recibe las predicciones de los clasificadores y las agrega de la manera que resulte más conveniente. Esta agregación se corresponde con el bloque **Fusion** de la Figura 17. Finalmente, tenemos la combinación de esos resultados agregados, representada por los rectángulos rojos y que se corresponden con el bloque **Interaction** de la Figura 17.

Si miramos la Figura 2, la Figura 11 y la Figura 18, vemos que el patrón que proponemos guarda semejanza con otros implementados en otros ámbitos con éxito, lo que nos permite reafirmar la viabilidad de nuestra propuesta de desarrollo.

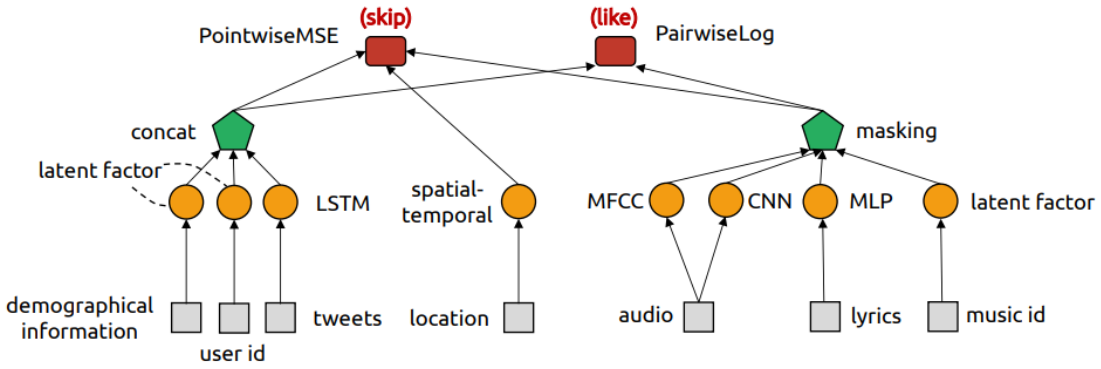

Figura 18. Esquema de funcionamiento OpenRec [37]

En segundo lugar, atendimos la problemática del formato. Una forma bastante común de expresar las emociones que se han detectado en un fichero multimedia es usar las seis emociones universales de Paul Ekman (alegría, tristeza, sorpresa, enfado, disgusto y sorpresa) además de una emoción «neutralidad» para indicar que no se ha detectado ninguna de las anteriores. Para cada emoción se indica el grado en que se ha detectado, o lo que es lo mismo, la **confianza** con la cual el clasificador ha detectado esa emoción en el medio que ha analizado.

El problema reside en que, por ejemplo, no todos los detectores usan la misma escala ni la misma distribución. Mientras que algunos utilizan valores relativos de 0 a 1, sumando todas las emociones 1 en total, otros utilizan escalas absolutas de 0 a 100, en las cuales cada emoción es independiente y la suma de todas ellas es mayor de 100. También hay casos como Beyond Verbal, que ni siquiera utiliza estas seis emociones básicas, sino que utiliza otras propias (véase subsección 3.2.3). Si además consideramos detectores como pulseras de actividad electrodérmica o de ritmo cardiaco, la diferencia entre resultados de los detectores se dispara. Es por esto por lo que se investigó de qué tipos de mecanismos disponíamos para clasificar las emociones.

Los modelos de clasificación de emociones pueden ser **categóricos** o **dimensionales** [38]. Según los enfoques de clasificación **categóricos**, las emociones se clasifican utilizando un conjunto de categorías finito. Ejemplos de clasificaciones categóricas serían algunos de los comentados anteriormente: las emociones detectadas se expresan en función del grado con el que se han detectado las seis emociones básicas universales.

- Alegría. 0.95
- Neutral. 0.04
- Tristeza. 0
- Miedo. 0.002
- Disgusto.
- Sorpresa. 0.008

Figura 19. Clasificación categórica de una lectura en la que se expresa alegría

Por el contrario, en los sistemas de clasificación **dimensionales** lo que hacemos es expresar una emoción como una tupla con tantos elementos como dimensiones distingamos. En la

Figura 20 podemos ver un ejemplo de una clasificación dimensional en la que se utilizan dos dimensiones, a saber, *valencia* y *excitación*. La valencia representa lo positiva o negativa que es una emoción, mientras que la excitación representa la energía que expresa esta. Para facilitar el entendimiento visual de estos sistemas de clasificación, suele hacerse una correspondencia entre emociones categóricas y las distintas dimensiones que haya, como es el caso de esta figura de ejemplo.

Aunque existen más dimensiones que pueden utilizarse para representar las emociones en un sistema dimensional, se considera que, usando la **valencia**, la **excitación** y la **dominancia** se tiene suficiente información para plasmar una emoción. Al sistema dimensional que utiliza estas tres dimensiones se le denomina **sistema PAD** (Pleasure, Agrado, Valencia; Arousal, Excitación; Dominance, Dominancia).

Si bien este tipo de clasificación permite un ajuste más detallado que las clasificaciones categóricas, al mismo tiempo presenta inconvenientes. Por ejemplo, el espacio PAD de tres dimensiones no es **isotrópico**, esto es, no mantiene las mismas propiedades en todas las direcciones como lo haría un sistema de coordenadas X, Y y Z, por lo que no se pueden aplicar distancias euclídeas, por ejemplo.

Para salvar este inconveniente, una solución intermedia que suele adoptarse es dividir ese espacio 3D en ocho sectores, distinguiendo para cada eje un sector negativo y uno

positivo (Figura 21). Así, en lugar de tener infinitas tuplas de tres elementos, solo tenemos ocho posibles combinaciones, que van desde (P-, A-, D-) a (P+, A+, D+).

Para adoptar un sistema dimensional, dado que la mayoría de los detectores de emociones utiliza clasificaciones categóricas, la pregunta a responder es si se puede hacer corresponder las emociones clasificadas de forma categórica con emociones clasificadas de forma dimensional. Según [38], llevar a cabo este mapeo es posible, puesto que pidieron a los participantes de su experimento que colocasen en el entorno 3D del sistema PAD un conjunto de emociones concretas, y el consenso fue bastante general, por lo que se aceptó la premisa de que las categorías de emociones pueden traducirse a espacios multidimensionales con cierto margen de error.

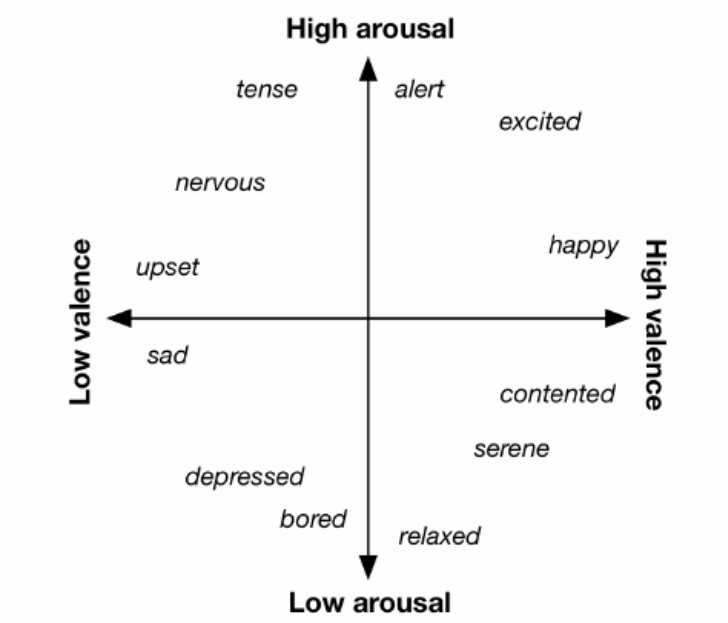

Figura 20. Clasificación dimensional de dos dimensiones [39]

Por todo lo anterior, decidió adoptarse PAD como el sistema de clasificación de las emociones, dejando en manos del desarrollador usuario de la API la forma en la que quiera realizar esa traducción entre sistemas de clasificación.

De cara a elegir la mejor forma de transmitir los resultados a los usuarios de la API se estudió cuál podría ser el mejor formato para utilizar. Si bien XML resulta más robusto cuando vamos a transmitir muchos datos con cierta complejidad, JSON es más ligero, menos redundante y más fácil de integrar en tecnologías basadas en *JavaScript*, por lo que todas las comunicaciones con la API se harán usando JSON como formato de las peticiones y respuestas [40].

En lo que respecta a la arquitectura, se ha optado por una arquitectura clásica basada en **cliente-servidor**. Tal y como se comentó en la sección sobre Consideraciones legales y

tecnológicas, la API se desarrollaría como una aplicación en la cual cada usuario, desarrollador y/o investigador despliega su propia instancia en la plataforma que considere. De implementarse la API de forma centralizada, habría que dedicar un gran esfuerzo de desarrollo a la creación de funcionalidad relacionada con la gestión de usuarios, de credenciales y de archivos multimedia; asegurar que la API cumple todos los requisitos que impone la AEPD en cuestión de tratamiento de datos de carácter personal; registrar el fichero correspondiente en la AEPD, etc.

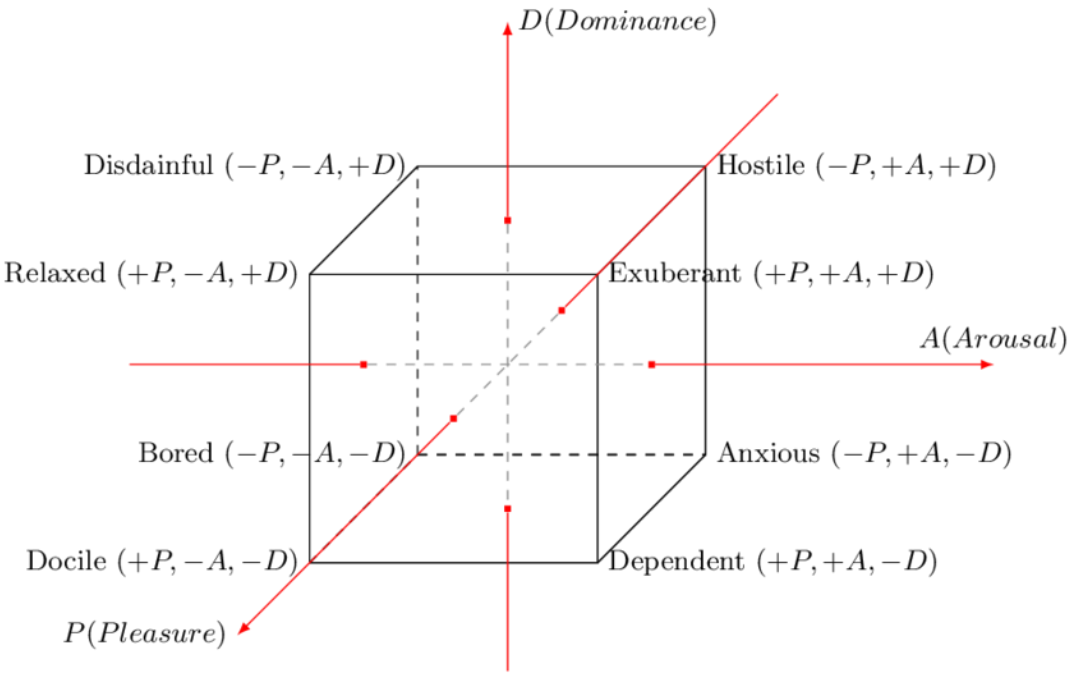

Figura 21. Clasificación dimensional PAD [41]

Finalmente, de cara a la futura implementación del sistema, este se definió de la siguiente manera.

- El framework define la *estructura lógica* que han de cumplir los detectores de emociones que se integren en el sistema. Así, en lugar de definir un conjunto cerrado de detectores, toda la API se programará a nivel de interfaz de manera que integrar una nueva tecnología sea cuestión de programar esa interfaz común.
- Con relación al punto anterior, la instancia de detectores se hará en base a un **fichero de configuración**, con la flexibilidad que esto nos aporta. El usuario desarrollador implementará la funcionalidad de los detectores que necesite y le indicará a la API los datos de cada detector mediante un fichero de configuración. En la Figura 26 (ANEXO A) podemos ver el formato de este fichero de configuración.

- El usuario ha de ser capaz de *filtrar* de alguna manera los detectores que va a usar, incluso si ya dispone de la flexibilidad que le aporta el fichero de configuración (véase punto anterior).
- El framework se encargará de *enviar los archivos multimedia* a los servicios de detección (si los hubiere). Se consideró la posibilidad de que la API recibiese directamente los resultados de cada detector, pero eso obligaba al usuario desarrollador a implementar el primer nivel de esa detección multimodal a dos niveles.
- El usuario desarrollador habrá de definir, como parte de esa estructura lógica, la forma en la que quiere que los resultados de un clasificador se *traduzcan al formato PAD*.
- Para dar mayor flexibilidad al desarrollador, se *conservarán los resultados originales*, así como las traducciones de los mismos en formato PAD, de manera que el usuario pueda comprobar posibles fallos en la traducción.
- La solicitud de análisis de un contenido multimedia cualquiera y la solicitud de los resultados de un análisis se hará con peticiones HTTP *distintas*. Dadas las características de la API (los servicios con los que hay que contactar suelen tener una latencia medianamente alta, la agregación de datos puede ser lenta si hay mucha información, etc.), preparar la API para que, ante una solicitud de análisis, respondiera con los resultados de analizar el contenido multimedia que fuese (Figura 30, ANEXO A) no tiene mucho sentido. Por este motivo, una solicitud de análisis solo tendrá como respuesta el código de estado que corresponda, en función de si se ha podido atender la petición o no. Para obtener los resultados de un análisis, habrá de hacerse una petición HTTP adicional.

En la Figura 31 (ANEXO A) podemos ver un diagrama de secuencia representando el funcionamiento general de la API. En el siguiente sprint se avanzará en el desarrollo del prototipo y se concretará el funcionamiento de cada parte del sistema.

4.3.3 Revisión

Debido a una planificación algo pobre (se subestimaron las tareas asociadas al desarrollo del prototipo), la historia de modelado del sistema ha quedado sin acabar: el modelado de integradores de detectores no ha sido terminado, por lo que esta historia vuelve al product backlog y se abordará en el siguiente sprint.

En lo que respecta al prototipo, se ha creado el repositorio base y se han hecho varias pruebas para comprobar el funcionamiento de ese primer prototipo. El funcionamiento de la API, que podemos ver representado en la Figura 31 (ANEXO A), se refinará en el siguiente

sprint. En este punto, la API recibe peticiones HTTP, las contesta y se han integrado dos tecnologías de detección a modo de pruebas de concepto, a saber, **Face++** y **Beyond Verbal**.

Tabla 5. Product backlog - Final Sprint 2

ID	Nombre	Descripción
S3	Modelado del sistema completo.	Como cliente quiero tener un sistema que sea capaz de integrar distintos canales de información de forma separada para luego integrar el resultado parcial de cada uno de ellos.

4.4 Sprint 3. Desarrollo del prototipo inicial

En este sprint se continuará el desarrollo del prototipo inicial, afianzando la arquitectura final del framework.

4.4.1 Planificación

Objetivo. Continuar y finalizar el desarrollo de ese prototipo inicial.

Duración. 31 días (27/11/20018 – 22/01/2019)

Durante este último y tercer sprint se decidió seguir avanzando en el desarrollo del prototipo que se empezó durante el sprint anterior. Dado que parte del trabajo de prototipado inicial se hizo usando servicios de terceros bajo licencias de evaluación, como parte de este sprint una de las tareas que se ha distinguido es la implementación de servicios de prueba (*mockup services*) de manera que, incluso si no se pudiese ver ver una agregación de datos reales, pudiésemos comprobar el funcionamiento de otras partes clave.

Además, como se comentó en la revisión del sprint anterior, quedó pendiente una tarea de la historia **S3**, sobre Modelado del sistema al completo, por lo que esta historia, que había vuelto al product backlog, pasa al sprint backlog de este sprint. En la Tabla 6 podemos ver el sprint backlog de este sprint. Tal y como se ha introducido, los esfuerzos de este sprint se enfocarán, por un lado, en acabar ese proceso de modelado que no se acabó en el sprint anterior, y, por otro lado, en perfeccionar el primer prototipo desarrollado.

A su vez, en la Tabla 7 podemos ver el desglose de tareas del sprint backlog. En este sprint lo que buscamos es definir los distintos puntos de acceso de la API, también llamados

endpoints. Además, durante este sprint se intentarán definir todas las clases y elementos que podamos necesitar a nivel lógico, incluso si se tuvieran que utilizar servicios de prueba.

Tabla 6. Sprint backlog - Sprint 3.

ID	Nombre	Descripción	SP
S3	Modelado del sistema completo.	Como cliente quiero tener un sistema que sea capaz de integrar distintos canales de información de forma separada para luego integrar el resultado parcial de cada uno de ellos.	5
S5	Completar prototipo inicial	Como cliente quiero que el prototipo previamente preparado se desarrolle hasta convertirse en una primera versión acabada del producto.	21
		Total	21

Tabla 7. Tareas de sprint backlog - Sprint 3.

Nombre	Tarea	Horas	+/-
Modelado del sistema completo.	Modelado de integradores de detectores	2	0
Completar prototipo inicial.	Determinar los *endpoints* de la API	4	0
	Preparar servicios para pruebas	3	0
	Implementar estrategias de fusión de datos	5	0
	Revisar técnicas de *bagging*	4	0
	Implementar asistente para subir contenido a un servidor	4	0
	Total	22	0

4.4.2 Desarrollo

Como se comentó en el sprint anterior, la API utiliza un fichero de configuración para instanciar los servicios de detección que hayan de iniciarse. En la Figura 26 (ANEXO A) podemos ver el formato de dicho fichero. Este fichero contiene una serie de elementos clave-valor, representando cada uno a un detector, siendo dicha clave el nombre o código identificador del mismo. Para cada detector se indica el *tipo de detección* de emociones que

hace (basado en la voz, en la expresión facial, etc.), el *tipo de ficheros multimedia* que soporta en su análisis, la *URL* a la que hay que dirigir la peticiones de análisis (si la hubiera), otros datos que pudiera necesitar el detector para funcionar (en algunos casos el servicio puede imponer un proceso de autenticación) y, por último, la ruta al fichero en la que se implementa esta *estructura lógica* común a todos los detectores.

El funcionamiento de la API comienza con la creación de los detectores en base al contenido de ese fichero. Para dar al usuario un grado más de control, se ha preferido dejar en manos del usuario la decisión de cuando realizar dicho proceso de creación. Así, el conjunto de funcionalidad que la API va a ofrecer a través de esos *endpoints* será el siguiente.

- **Inicio de la API**. Aunque el servidor de la API esté desplegado, los detectores de la misma no se iniciarán hasta que el usuario no haga una petición para ello. Se consideró la idea de hacer esto automáticamente al lanzar el servidor, pero eso equivaldría a realizar una acción de forma fija, limitando al usuario de la API.
- **Filtrado de resultados**. Además de mediante la configuración del fichero, el usuario puede estar interesado en disponer de un punto de entrada a la API para modificar los detectores que va a utilizar una vez la API ya se ha iniciado.
- **Análisis de multimedia**. La actividad clave de la API consiste en el análisis de multimedia, por lo que se hace imprescindible un punto de acceso para llevar a cabo esta actividad. Puesto que la aportación principal de la API consiste en la integración de resultados, el punto de acceso para el análisis simplemente recibirá la información multimedia que haya que analizar y la API se encargará de distribuirla a los detectores y de integrar los resultados posteriormente.
- **Recuperación de resultados**. La actividad de la API estará fuertemente afectada por la latencia de los servicios que integra, por lo que replicar aquí el enfoque que siguen estos servicios (una petición de análisis se responde con los resultados) no resulta práctico ni lógico, dada la naturaleza del sistema. Por tanto, la recuperación de resultados se hará con un punto de acceso distinto al utilizado para solicitar un análisis.

Así, esta funcionalidad, teniendo en cuenta esas circunstancias, ha quedado plasmada de la siguiente manera.

- **`/init` -** Crea los detectores tal y como están indicados en el fichero de configuración y realiza una actividad de *benchmarking* inicial para medir los tiempos de respuesta cada detector. Gracias a este proceso, que puede ser tan exhaustivo como el desarrollador quiera, la API consigue calcular dinámicamente un atributo `delay`, que expresa en milisegundos el tiempo que tarda el detector, de media, en obtener una respuesta.

- **`/setup`** - Una de las funcionalidades que se estimó como valiosa fue la capacidad de filtrar los detectores en uso una vez que la API ha sido iniciada. Para cubrir dicha funcionalidad se utiliza este *endpoint*. Este punto de acceso nos permite filtrar los detectores que van a utilizarse en base al canal afectivo que analicen (cara, voz, cuerpo, actividad electrodérmica, etc.), a su tiempo de respuesta (atributo `delay`) y a si funcionan en tiempo real o no (atributo `realTime`).
- **`/analyse`** - Este *endpoint* nos ofrece la funcionalidad central de la API, a saber, *la detección y el análisis de emociones*. A través de este punto de acceso la API recibe el contenido multimedia que haya que analizar junto con dos parámetros de configuración, indicando que tipo de información hay que buscar y qué tipo de contenido se refleja en ese contenido multimedia.
- **`/results`** – A través de este *endpoint* recibiremos los resultados que haya acumulado hasta el momento la aplicación. Aunque los resultados se devuelven de forma acumulativa, en la misma respuesta se contesta con los resultados individuales de cada detector y los resultados parcialmente agregados de cada conjunto de detectores, todo ello usando el formato PAD.
- **`/results-raw/`** - Este *endpoint* tiene una funcionalidad similar al anterior, salvo que en este caso se devuelven todos los datos individuales generados por cada detector en el formato propio de cada uno. Una vez más, esta funcionalidad se añade en pos de la flexibilidad de la API, dando al usuario la capacidad de ver los datos en bruto que la API ha recopilado.

Dado que el sistema planteado se asemeja mucho a un *ensembler* de clasificadores, se estudiaron distintas formas de realizar esa fase de votación que se realiza en las técnicas de *bagging*. Fruto de este estudio surgió una idea para aplicar la clasificación automática para añadir un nivel más de confianza a nuestra clasificación, pero como escapa del alcance de este TFM, se plantea explorar esa idea durante el desarrollo de la tesis doctoral.

4.4.3 Revisión

Al acabar este sprint, disponíamos de una API implementada en Node.js que se configura de forma dinámica con un fichero de configuración, que permite realizar un filtrado de los servicios utilizados en base a características de los mismos, que hace peticiones dinámicamente a los servicios que corresponda cuando recibe un fichero multimedia y que es capaz de agregar datos fruto de esos análisis y devolverlos al cliente.

Las técnicas de *bagging* se dejaron para trabajo futuro, como se comentará más adelante, pues implicaría realizar una traducción de un sistema dimensional (PAD) a un sistema categórico que pueda usarse en un proceso de clasificación automática.

Dadas las fechas que ha alcanzado el trabajo y que este forma parte de una futura tesis, se decide parar aquí el desarrollo, siendo este el último sprint realizado.

4.5 Conclusiones

En este capítulo se ha presentado el desarrollo de la propuesta de este TFM siguiendo la metodología Scrumban. Se hicieron algunas adaptaciones en base a las circunstancias del proyecto y se intentó ceñir el desarrollo a la planificación realizada en el anteproyecto.

No obstante, el desarrollo real dista bastante del desarrollo que se planificó en septiembre. En la Tabla 8 podemos ver una comparativa de los sprints que se propusieron en el anteproyecto contra los que han resultado realmente. Los sprints reales tienen una duración media de un mes, doblando la duración media de los sprints planificados en el anteproyecto.

Tabla 8. Visión retrospectiva de los sprints realizados

Sprints estimados	Fecha de fin	Sprints realizados	Fecha de fin
Revisión de la literatura	08/10/2018	Revisión sistemática de la literatura	22/10/2018
Análisis de tecnologías de mayor impacto	22/10/2018		
Modelado de detectores multimodales	26/11/2018	Modelado del sistema multimodal y prototipo inicial	26/11/2018
Combinación de detectores del mismo tipo	17/12/2018		
Estudio de resultados	14/01/2019	Desarrollo del prototipo inicial	22/01/2019
Validar el sistema	31/01/2019		

El trabajo comenzado en este TFM supone en realidad el principio de una tesis doctoral, y por tanto constituye la base sobre la que continuar trabajando en esa línea.

CAPÍTULO 5. DESCRIPCIÓN DE LA APLICACIÓN

En este capítulo se hará una descripción completa del prototipo desarrollado. Se analizarán las distintas formas de desplegarlo, las estructuras usadas para desarrollarlo y los distintos mecanismos a través de los cuales personalizar la API para nuestras circunstancias y/o necesidades particulares. Los diagramas referidos en este capítulo pueden encontrarse en el ANEXO A.

5.1 Introducción

La propuesta inicial del presente trabajo consistía en un marco de trabajo o framework que permitiese a desarrolladores e investigadores integrar de forma mucho más sencilla la detección de emociones en sus aplicaciones. Pero no solo eso, sino que dicho framework, desarrollado en forma de API, permitiría integrar de forma automática los resultados producidos por distintos servicios, aplicando un proceso de normalización a resultados expresados en formatos dispares.

La aportación principal de esta API no radica únicamente en su arquitectura modular y extensible, sino en el hecho de que hace totalmente transparente la detección multimodal de emociones, una de las formas de detección de emociones más potentes y precisas. Incluso cuando esto es un hecho reconocido, durante años los desarrolladores de aplicaciones han estado evitando la implementación de estos sistemas multimodales, debido precisamente al incremento de la dificultad que añade al desarrollo.

En la Figura 22 podemos ver un ejemplo de lo que supondría actualmente desarrollar una aplicación o sistema que integrase la detección de emociones en su funcionamiento. Incluso cuando los servicios de detección de emociones de terceros ahorran mucho trabajo a los desarrolladores, puesto que, de no tenerlos, tendrían que implementar los suyos propios, estos aún deben enfrentar desafíos. Por ejemplo, aún deben implementar la comunicación con los servicios (o integrar la SDK de los mismos en el proyecto, lo que puede suponer un problema cuando un servicio no ofrece su API y no existe SDK para la plataforma en la que los desarrolladores estén trabajando), realizar un proceso de interpretación exclusivo para cada servicio, pues cada uno utiliza un formato propio, etc.

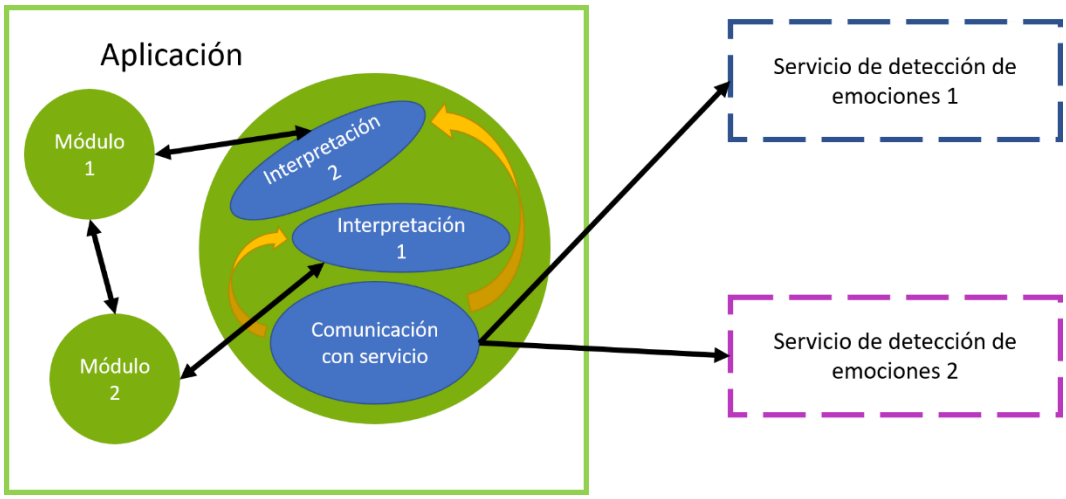

Figura 22. Aplicación con enfoque clásico

Con el framework desarrollado en el contexto de este trabajo, esas dificultades desaparecen. En la Figura 23 podemos ver un ejemplo de lo que supondría desarrollar la misma aplicación anterior, pero usando el framework propuesto. Una vez el desarrollador ha configurado la API mediante el fichero de configuración (Figura 26, ANEXO A), esta queda preparada. Ante peticiones de detección de emociones, la API toma como entrada un fichero multimedia, así como una serie de parámetros para indicar el tipo de fichero que es y el tipo de información afectiva que contiene. De forma totalmente transparente, la API se encarga de redirigir la petición a los detectores de emociones pertinentes para, posteriormente, realizar una agregación de los resultados y devolverlos al cliente.

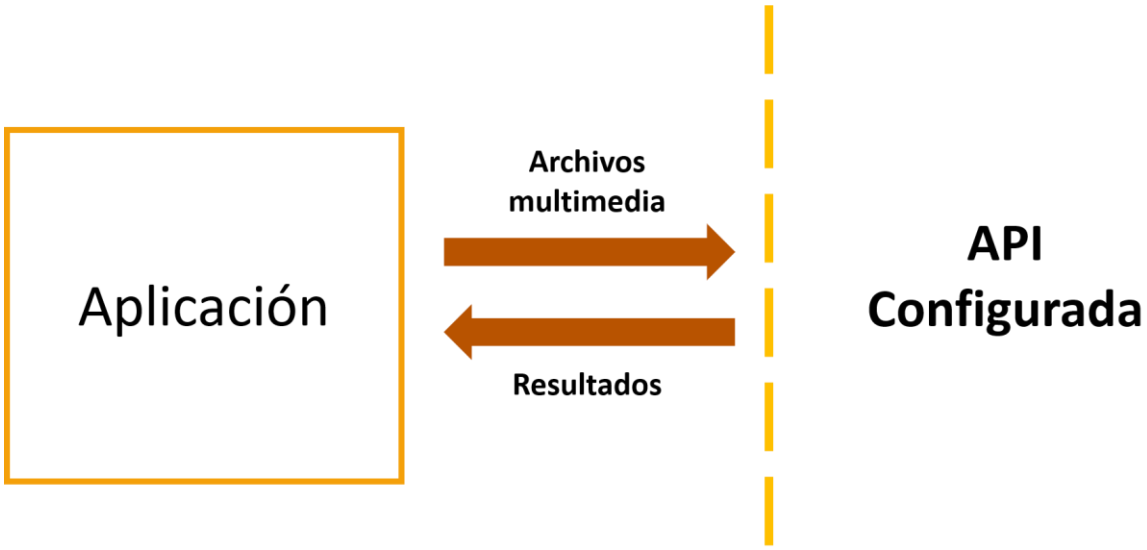

Figura 23. Aplicación usando la API propuesta

De esta manera, cada una de las peticiones HTTP individuales que los desarrolladores tenían que hacer antes, se reduce a una sola, simplificándose también el proceso de interpretación, al seguir los resultados un único formato.

Otra de las aportaciones que hace este framework al estado del arte de la detección multimodal es la integración de datos a dos niveles que se realiza. Internamente, la API organiza los detectores de emociones en base al canal afectivo al que atienden, ya sea la expresión facial, la voz, la postura, la actividad eléctrica del cerebro, etc. Así, por ejemplo, una expresión facial no es leída por un solo servicio, sino por varios, lo que permite realizar una validación cruzada *solo* a nivel de expresión facial para detectar posibles errores de interpretación y/o afianzar el resultado de un solo canal, sin que el análisis de esa expresión facial se vea afectado por los resultados de otros canales. Al hacer esto para cada canal, disponemos de una serie de resultados mucho más sólidos y fiables que volvemos a fusionar siguiendo la misma metodología. Esta fusión de datos a dos niveles nos permite generar un único resultado que ha sido doblemente validado, y por tanto, con garantías de ser correcto.

5.2 Requisitos y recomendaciones de uso

En cuestión de uso del framework, al ofrecerse en forma de API, puede usarse en cualquier circunstancia en la que se disponga de conexión a Internet, puesto que no se necesita ninguna capacidad de cómputo especial para hacer peticiones HTTP. En cuestión de despliegue, el equipo en el que se despliegue la API ha de cumplir ciertos requisitos:

- Node.js en su versión 10.15.0.
- Accesible desde equipo de Internet (si fuera necesario).
- Conexión a Internet.
- Intel Core i7 @ 2.50 GHz.
- 8 Gb de memoria RAM.

Los dos últimos requisitos reflejan las características del equipo en el que se hizo el desarrollo de la API, pero no suponen ninguna cota superior o inferior de requisitos de cara a desplegar la API en otro equipo.

Por último, es importante considerar que la API, una vez desplegada y en uso, ha de cumplir las restricciones que imponga la ley de protección de datos que corresponda a la zona geográfica en la que esta se encuentre desplegada.

El código fuente de este framework puede encontrarse en GitHub [42] bajo la licencia BSD de 3 clausulas.

5.3 Estructura de la aplicación y caso práctico

En la Figura 29 podemos ver un diagrama de componentes del framework desarrollado, llamado ***Tot System***. Las partes más importantes son el *DetectorHandler* y la interfaz *Detector*. Esto no ha de restar importancia al resto de módulos de la aplicación, por supuesto, pero son estas dos primeras clases las que encapsulan la modularidad y extensibilidad del sistema.

Gracias a los campos indicados en el fichero de configuración, el componente *DetectorHandler* es capaz de separar y agrupar los distintos tipos de detectores en base a lo que son capaces de detectar. A su vez, al estar el resto de funcionalidad del sistema programado a nivel de interfaz, agregar nuevos detectores al sistema tan solo requiere implementar la interfaz definida en el prototipo *Detector*, que podemos observar en la Figura 27.

Gracias a esta flexibilidad y modularidad, la API puede conocer tantos sistemas de detección de emociones como los usuarios de la misma necesiten. Actualmente, por

cuestiones de disponibilidad, las pruebas que se han hecho solo han involucrado tecnologías que ofrecían sus servicios a través de peticiones HTTP, pero el framework no se limita solo a eso. Al estar implementado a nivel de interfaz, las instancias de *Detector* pueden encapsular cosas muy distintas. Por ejemplo, si un desarrollador quisiese integrar datos provenientes de sensores que sus usuarios tienen distribuidos por el cuerpo (*wearables*, como una pulsera de actividad o *smartwatch*), podría desplegar la API de forma local y completar el código de la función *extractEmotions* con el código necesario para conectarse por Bluetooth o por el protocolo correspondiente a esos sensores.

Caso práctico

Supongamos que un desarrollador ha implementado un juego serio para aprender inglés pensado para niños. Dicho juego incluye diversos tipos de actividades para medir el nivel de inglés de sus usuarios en sus distintas vertientes: comprensión lectora, conocimientos de vocabulario, etc. En la Figura 24 y en la Figura 25 podemos ver sendos prototipos de esta aplicación ficticia.

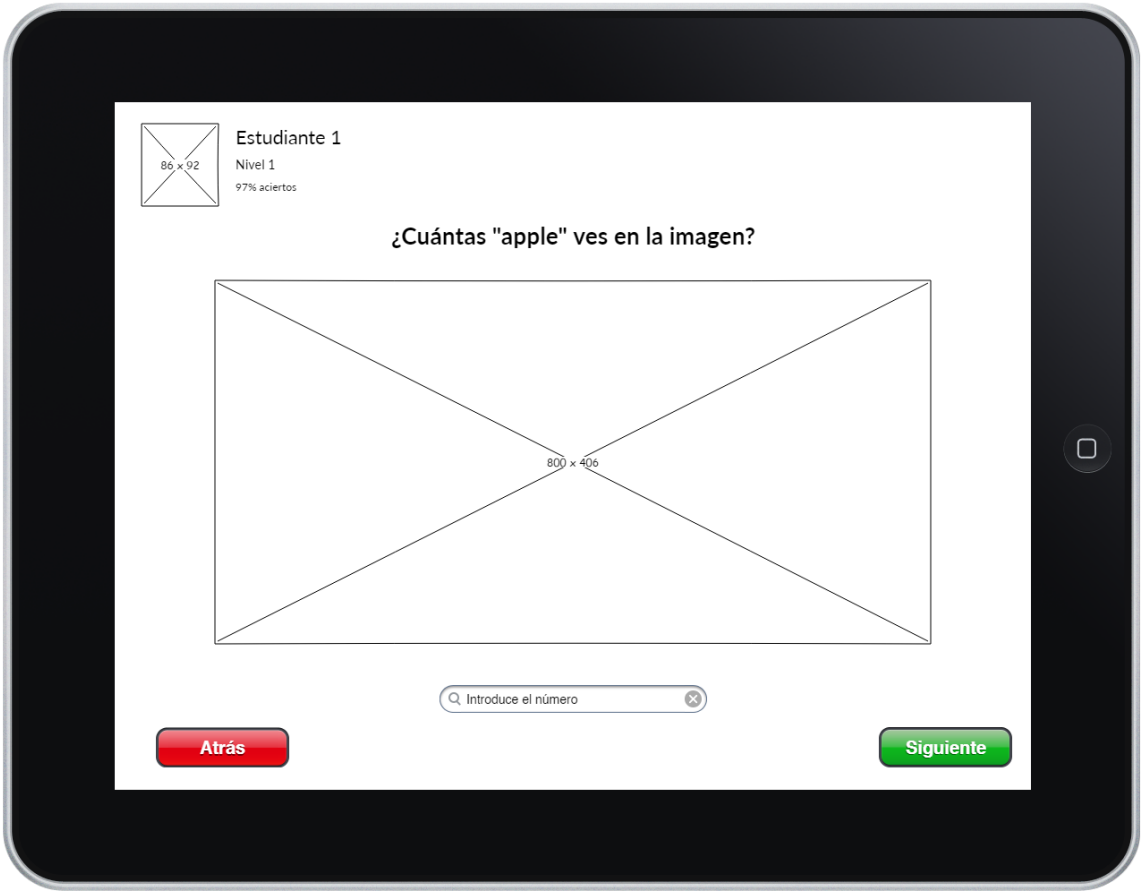

Figura 24. Maqueta de ejemplo – Primera pantalla

Ahora bien, el desarrollador de la misma sabe que, de no controlar el sobreesfuerzo cognitivo que supone para el usuario el uso de la aplicación, una experiencia que debería resultar amena se convierte en una situación frustrante, perdiéndose todos los efectos positivos que un juego serio como ese podría haber aportado al proceso de aprendizaje.

Figura 25. Maqueta de ejemplo – Segunda pantalla

El desarrollador conoce la existencia de tecnologías que permiten conocer el estado emocional del usuario, cosa que podría suponer una solución a sus problemas, pero no tiene conocimientos sobre Computación Afectiva: no sabe qué tecnologías podría usar, no sabe cómo podría integrarlas, si es que puede integrarlas en su plataforma, etc. Este desarrollador ficticio se ha encontrado con el principal problema que experimentan aquellos que consideran integrar la Computación Afectiva en sus proyectos: se dispone de una gran cantidad de información y de una gran cantidad de sistemas distintos, todos ellos funcionando de forma distinta, lo que supone una dificultad muy grande para el desarrollador.

No obstante, ese desarrollador se encuentra nuestra API. Al estar organizada por tipos de detectores, como podemos ver en la estructura de directorios de la Figura 36, el desarrollador puede elegir los detectores que quiere y/o puede usarla dejándose guiar por el

sistema de directorios. Usando la documentación de la API [27] , el desarrollador adquiriría las licencias que necesitase y elaboraría su propio fichero de configuración. Dado que su aplicación se utiliza en una tableta e incluye actividades de dictado, el desarrollador solo tiene que implementar esa recogida de información y las correspondientes peticiones HTTP a la API. En este punto, el desarrollador solo tendría que buscar la información acerca del sistema de clasificación PAD para interpretar los resultados de la API.

Gracias a la API desarrollada, ese usuario ficticio ha podido superar los primeros obstáculos que impiden a muchos la entrada en el mundo del desarrollo de aplicaciones afectivas. En este punto, y dada la licencia con la que está distribuida la API, el desarrollador podría enriquecer sus mecanismos de agregación, añadir soporte para nuevos detectores, etc., contribuyendo al crecimiento de este proyecto.

CAPÍTULO 6. CONCLUSIONES Y PROPUESTAS

Por último, en este capítulo se presentan las conclusiones extraídas de la elaboración de este trabajo, así como una revisión de los conocimientos adquiridos durante el desarrollo del mismo. Como se comentó en otros capítulos, este trabajo está enmarcado en el desarrollo de una tesis doctoral, por lo que también se recogen algunas propuestas para el trabajo futuro.

6.1 Conclusiones

El proyecto de este TFM supone el comienzo de una de las líneas de investigación de una tesis doctoral. El TFG que precede a este trabajo también se encontraba encaminado al campo de la investigación y a la Computación Afectiva, pero este suponía una prueba de concepto. En este trabajo se ha consolidado el conocimiento en el campo de investigación de la Computación Afectiva, permitiendo detectar lagunas y carencias y llegar a proponer soluciones para cubrir algunas de las lagunas detectadas en este campo.

En el marco de este trabajo se realizó una revisión sistemática con una finalidad doble. Por un lado, se pretende crear una base de conocimiento sólido que permitiese justificar la razón de este trabajo. Por otro lado, se persigue identificar, de una forma sistemática, el estado del arte en cuestión de la aplicación de la Computación Afectiva. Al recoger de forma sistemática qué es lo que se está haciendo en esta disciplina, estamos conociendo también qué no se está haciendo. Con este trabajo, y con los futuros trabajos que se hagan enmarcados en la tesis doctoral, se pretende cubrir algunas de estas lagunas, o al menos realizar propuestas sólidas para cubrirlas.

Una de esas lagunas era la poca aplicación que había de la Computación Afectiva en entornos reales, frente a los recursos y esfuerzos que están dedicándose a la mejora de detectores de emociones individuales. La idea que se extrae de los artículos revisados durante la revisión sistemática, así como artículos revisados fuera de la misma, es clara: la detección de emociones multimodal es la mejor opción, la más precisa y la que más información aporta. No obstante, su correcta implementación introduce problemáticas en el desarrollo que los investigadores prefieren evitar.

Como propuesta para solucionar este problema surgió la idea de este TFM. En este trabajo hemos propuesto un marco de trabajo dirigido a desarrolladores, que les permita, de forma fácil y transparente, la integración de detectores de emociones en sus proyectos. De manera que ese techo de cristal que supone integrar emociones en el comportamiento de una aplicación se vaya rompiendo gradualmente.

Dicho marco de trabajo, desarrollado en forma de API, supone el principio de un proyecto que se continuará más allá de este TFM, por lo que el sistema desarrollado hasta ahora no está cerrado y seguirá creciendo en el futuro. De hecho, se ha diseñado de forma flexible para permitir integrar nuevas técnicas y tecnologías de reconocimiento de emociones que puedan surgir en un futuro.

Para facilitar su uso y su integración experimental, el código de la aplicación, junto con una extensa documentación, se encuentra en Github con una licencia BSD de tres clausulas [42]. Dada su arquitectura y extensibilidad, la API puede enriquecerse mucho, por ejemplo, con aportaciones de otros desarrolladores implementando la interfaz Detector para los distintos servicios de detección de emociones que pueden hallarse en la red.

6.2 Trabajo futuro y posibles ampliaciones

Tal y como se ha indicado anteriormente, el sistema aquí desarrollado no es un producto cerrado, y la funcionalidad del mismo se podrá ir adaptando en el tiempo según las necesidades. En este punto, algunas de las propuestas de trabajo futuro a implementar en el sistema son las que siguen:

Agrupamiento de detecciones

Actualmente la API realiza una agregación por tipo de detector primero, y de forma global después. No obstante, podría ser interesante realizar esta agregación de forma más localizada, asociando, por ejemplo, detecciones concretas para que se agreguen juntas, descartándose otras en el proceso.

Traducción a sistema categórico

Como un mecanismo de asistencia más para esos desarrolladores usuarios que desconocen aspectos teóricos de la Computación Afectiva, podría ofrecerse un resultado final que haya sido expresado mediante una categoría de un conjunto finito (como puede ser «alegría», «aburrimiento», etc.), en lugar de usar los valores del sistema dimensional PAD. Si bien el sistema PAD permite expresar emociones de forma más detallada, un conjunto de tres números no resulta significativo para un desarrollador primerizo en el campo de la Computación Afectiva. Por ese mismo motivo, traducir esos valores PAD a una categoría concreta abre la puerta al uso de categorías más personalizadas, más alejadas de las categorías estándares habituales.

Clasificador automático de categorías

Si se obtuviese como producto final una etiqueta de entre un conjunto de ellas, podría prepararse un clasificador automático que, antes ciertas tuplas de valores PAD de distintos detectores predijese una de esas categorías. Con un conjunto de entrenamiento lo suficientemente grande, esto supondría una tercera línea de validación en el sistema multimodal de dos niveles que se ha desarrollado.

Técnicas de agregación

Actualmente, la propuesta de la API realiza una fusión de datos relativamente sencilla, una agregación de resultados basada en calcular la media aritmética de cada componente de las tuplas en formato PAD que representan los resultados. Un proceso de agrupación más complejo (asignando pesos a las distintas tuplas, o a las tuplas de detectores concretos), podría ofrecer una visión distinta de unos mismos resultados.

Validación de la API

Es importante hacer pruebas finales para verificar la fiabilidad, usabilidad, etc. del framework propuesto. Cuando la API haya madurado lo suficiente, se implementarán los tipos de prueba más adecuados para este tipo de sistemas.

BIBLIOGRAFÍA

[1] R. W. Picard, «Affective Computing. Tech. Rep. 321,» MIT Media Lab, 1995.

[2] Gartner, «Gartner,» 16 Agosto 2016. [En línea]. Available: http://www.gartner.com/newsroom/id/3412017.

[3] R. A. Calvo y S. D'Mello, «Affect detection: An interdisciplinary review of models, methods, and their applications,» *IEEE Transactions on Affective Computing*, vol. 1, nº 18-37, p. 1, 2010.

[4] J. M. Garcia Garcia, Variación Dinámica del Comportamiento de las Aplicaciones en Función de las Emociones de Usuario, Albacete: Escuela Superior de Ingeniería Informática, Trabajo de Fin de Grado, 2017.

[5] I. Morgun, «Types of machine learning algorithms,» 24 Diciembre 2015. [En línea]. Available: http://en.proft.me/2015/12/24/types-machine-learning-algorithms/. [Último acceso: 27 Noviembre 2018].

[6] D. Budgen y P. Brereton, «Performing systematic literature reviews in software engineering,» de *Proceeding of the 28th international conference on Software engineering - ICSE '06*, 2006.

[7] ACM, «ACM Digital Library,» Association for Computing Machinery, [En línea]. Available: https://dl.acm.org/. [Último acceso: 19 Diciembre 2018].

[8] M. P. Ph.D., «The Benefits of Emotional Awareness,» Psychology Today, 5 Enero 2018. [En línea]. Available: https://www.psychologytoday.com/us/blog/between-cultures/201801/the-benefits-emotional-awareness. [Último acceso: 10 Enero 2019].

[9] Black Dog Institute, «Black Dog Institute,» Black Dog Institute, [En línea]. Available: https://www.blackdoginstitute.org.au/. [Último acceso: 16 Enero 2019].

[10] L. De Silva, T. Miyasato y R. Nakatsu, «Facial emotion recognition using multi-modal information,» *Proceedings of ICICS, 1997 International Conference on Information, Communications and Signal Processing. Theme: Trends in Information Systems Engineering and Wireless Multimedia Communications (Cat. No.97TH8237)*, vol. 1, pp. 397-401, 1997.

[11] A. Jaimes y N. Sebe, «Multimodal human–computer interaction: A survey,» *Computer Vision and Image Understanding*, vol. 108, pp. 116-134, 2007.

[12] E. Lutins, «Ensemble Methods in Machine Learning: What are They and Why Use Them?,» Towards Data Science, 2 Agosto 2017. [En línea]. Available:

https://towardsdatascience.com/ensemble-methods-in-machine-learning-what-are-they-and-why-use-them-68ec3f9fef5f. [Último acceso: 23 Enero 2019].

[13] P. Varhol, «To agility and beyond: The history—and legacy—of agile development,» TechBeacon, [En línea]. Available: https://techbeacon.com/agility-beyond-history%E2%80%94-legacy%E2%80%94-agile-development. [Último acceso: 18 Enero 2019].

[14] Learning Tree, «Is Your Project Plan-Driven or Change-Driven?,» 12 Noviembre 2012. [En línea]. Available: http://blog.learningtree.com/is-your-project-plan-driven-or-change-driven/. [Último acceso: 18 Enero 2019].

[15] K. Beck, M. Beedle, A. v. Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland y D. Thomas, «Manifiesto por el Desarrollo Ágil de Software,» 2001. [En línea]. Available: http://agilemanifesto.org/iso/es/manifesto.html. [Último acceso: 18 Enero 2019].

[16] «Comprehensive Guide to the Agile Manifesto,» Smartsheet, [En línea]. Available: https://www.smartsheet.com/comprehensive-guide-values-principles-agile-manifesto. [Último acceso: 18 Enero 2019].

[17] «Kanban,» Wikipedia, [En línea]. Available: https://es.wikipedia.org/wiki/Kanban. [Último acceso: 21 Enero 2018].

[18] «¿Por qué utilizar la metodología Kanban? Método básico y principios Kanban para ayudarle a empezar,» kanban tool, [En línea]. Available: https://kanbantool.com/es/metodologia-kanban. [Último acceso: 21 Enero 2019].

[19] «Kunagi,» Kunagi, [En línea]. Available: http://kunagi.org/. [Último acceso: 21 Enero 2018].

[20] Trello, «Trello,» Trello, [En línea]. Available: https://trello.com/. [Último acceso: 21 Enero 2019].

[21] «About Node.js®,» [En línea]. Available: https://nodejs.org/en/about/. [Último acceso: 22 Enero 2019].

[22] «Express. Infraestructura web rápida, minimalista y flexible para Node.js,» [En línea]. Available: https://expressjs.com/es/. [Último acceso: 25 Enero 2019].

[23] Face++, «Face++ Cognitive Services,» Face++, [En línea]. Available: https://www.faceplusplus.com/. [Último acceso: 23 Enero 2019].

[24] Beyond Verbal, Beyond Verbal, [En línea]. Available: https://beyondverbal.com/. [Último acceso: 22 Enero 2019].

[25] Postman, «Postman,» [En línea]. Available: https://www.getpostman.com/. [Último acceso: 24 Enero 2019].

[26] «Visual Studio Code,» Microsoft, [En línea]. Available: https://code.visualstudio.com/?wt.mc_id=DX_841432. [Último acceso: 25 Enero 2019].

[27] J. M. García García, «JSDoc Home Documentacion,» [En línea]. Available: https://josegarciaclm95.github.io/tot-system/docs/v1/. [Último acceso: 25 Enero 2019].

[28] Noticias Juricias, «Real Decreto 1720/2007, de 21 de diciembre, por el que se aprueba el Reglamento de desarrollo de la Ley Orgánica 15/1999, de 13 de diciembre, de protección de datos de carácter personal,» [En línea]. Available: http://noticias.juridicas.com/base_datos/Admin/rd1720-2007.t1.html. [Último acceso: 25 Enero 2019].

[29] tenstep, «SCRUM: Cómo escribir historias de usuarios sin morir en el intento,» [En línea]. Available: https://www.tenstep.ec/portal/articulos-boletin-tenstep/41-scrum/253-scrum-como-escribir-historias-de-usuarios-sin-morir-en-el-intento. [Último acceso: 25 Enero 2019].

[30] El Laboratorio de las TI, «Método de Estimación Ágil: Puntos de Historia,» [En línea]. Available: http://www.laboratorioti.com/2013/02/21/metodo-de-estimacion-agil-puntos-de-historia/. [Último acceso: 25 Enero 2019].

[31] T. Dybå y T. Dingsøyr, «Empirical studies of agile software development: A systematic review,» *Information and Software Technology*, vol. 50, nº 9-10, pp. 833-859, 2008.

[32] M. Jorgensen y M. Shepperd, «A Systematic Review of Software Development Cost Estimation Studies,» *IEEE Transactions on Software Engineering*, vol. 33, nº 1, pp. 33-53, 2007.

[33] «Matplotlib,» Matplotlib , [En línea]. Available: https://matplotlib.org/. [Último acceso: 25 Enero 2019].

[34] «Python Data Analysis Library,» Pandas, [En línea]. Available: https://pandas.pydata.org/. [Último acceso: 25 Enero 2019].

[35] «NumPy,» NumPy, [En línea]. Available: http://www.numpy.org/. [Último acceso: 25 Enero 2019].

[36] L. Yang, «OpenRec,» [En línea]. Available: http://openrec.ai/. [Último acceso: 25 Enero 2019].

[37] L. Yang, E. Bagdasaryan, J. Gruenstein, C.-K. Hsieh y D. Estrin, «OpenRec,» [En línea]. Available: http://openrec.ai/. [Último acceso: 25 Enero 2019].

[38] H. Hoffmann, A. Scheck, T. Schuster, S. Walter, K. Limbrecht, H. C. Traue y H. Kessler, «Mapping discrete emotions into the dimensional space: An empirical approach,» *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, nº Octubre, pp. 3316-3320, 2012.

[39] D. Graziotin, X. Wang y P. Abrahamsson, «Understanding the Affect of Developers: Theoretical Background and Guidelines for Psychoempirical Software Engineering,» *7th Intl. Workshop on Social Software Engineering,* pp. 25-32, 2015.

[40] K. Mikoluk, «JSON vs XML: Cómo JSON es Superior a XML,» Udemy, 31 Diciembre 2013. [En línea]. Available: https://blog.udemy.com/json-vs-xml-como-json-es-superior-a-xml/. [Último acceso: 25 Enero 2019].

[41] S. Tarasenko, «Emotionally Colorful Reflexive Games,» 2010.

[42] J. M. García García, «Tot System,» [En línea]. Available: https://github.com/josegarciaclm95/tot-system. [Último acceso: 25 Enero 2019].

[43] J. Gonzalez-Sanchez, M.-E. Chavez-Echearagay, R. Atkinson y W. Burleson, «Affective computing meets design patterns: A pattern-based model for a multimodal emotion recognition framework,» *Proceedings of the 16th European Conference on Pattern Languages of Programs - EuroPLoP '11,* pp. 1-11, 2011.

[44] J. Tao y T. Tan, «Affective Computing: A Review,» *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics),* vol. 3784 LNCS, pp. 981-995, 2005.

[45] A. Nightingale, «A guide to systematic literature reviews,» *Surgery (Oxford),* vol. 27, nº 9, pp. 381-384, 2009.

[46] Wikipedia, «Systematic review,» [En línea]. Available: https://en.wikipedia.org/wiki/Systematic_review. [Último acceso: 12 Diciembre 2018].

[47] IEEE, «IEEE Transactions on Affective Computing,» 2019. [En línea]. Available: https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=5165369. [Último acceso: 7 Enero 2019].

[48] M. Oehl, F. W. Siebert, T.-K. Tews, R. Höger y H.-R. Pfister, «Improving human-machine interaction - A non invasive approach to detect emotions in car drivers,» *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics),* vol. 6763 LNCS, nº 3, pp. 577-585, 2011.

[49] S. Pahuja, «What is Scrumban?,» Agile Alliance, [En línea]. Available: https://www.agilealliance.org/what-is-scrumban/. [Último acceso: 21 Enero 2019].

[50] J. M. Garcia-Garcia, V. M. R. Penichet y M. D. Lozano, «Emotion detection: a technology review,» de *Proceedings of the XVIII International Conference on Human Computer Interaction - Interacción '17*, México, 2017.

[51] «Getting Started with JSDoc 3,» [En línea]. Available: http://usejsdoc.org/about-getting-started.html. [Último acceso: 25 Enero 2019].

[52] Á. López, «DISTRIBUTED APPLICATIONS AND NODE.JS,» 15 Marzo 2014. [En línea]. Available: https://ajlopez.github.io/Talks/NodeDistributedApps/index.html#/. [Último acceso: 25 Enero 2019].

CONTENIDO DEL CD

En el contenido del CD que acompaña a la memoria podemos encontrar los siguientes recursos:

- Memoria del trabajo en los formatos PDF, DOCX y DOC dentro del directorio Memoria.
- Código fuente del trabajo dentro del directorio Código fuente.
- Libros y artículos a los que se ha hecho referencia durante la memoria y que se han utilizado como bibliografía. Los cuales podemos encontrar en el directorio Bibliografía.

ANEXO A. **DIAGRAMAS UML Y DOCUMENTACIÓN**

En este anexo se recogen los artefactos UML que se han generado durante el desarrollo del prototipo. En este anexo pueden encontrarse los siguientes artefactos:

- Diagramas de clases.
- Diagrama de secuencia. Se presenta la cadena de peticiones entre entes que suponen las distintas peticiones HTTP para las que ofrece soporte la API.
- Diagrama de componentes.
- Estructura de directorios del sistema.

Ejemplo de fichero de configuración

```
{
    "nombre-detector": {
        "category": "face/voice/body/eda/...",
        "media": [ "tipos", "de", "archivos", "multimedia", "que",
"el", "detector", "puede", "usar", "en", "su", "analisis"],
        "realTime": true/false,
        "url": "https://url-del-servicio-si-lo-hubiera",
        "otherOptions": {
            "clave": "este atributo se utiliza para datos como la
clave de inicio de sesión del servicio",
        },
        "callbacks":
"./fichero/en/el/que/se/define/la/funcionalidad/del/detector.js"
    }
}
```

Figura 26. Formato de fichero de configuración

Diagrama de clases – Detector

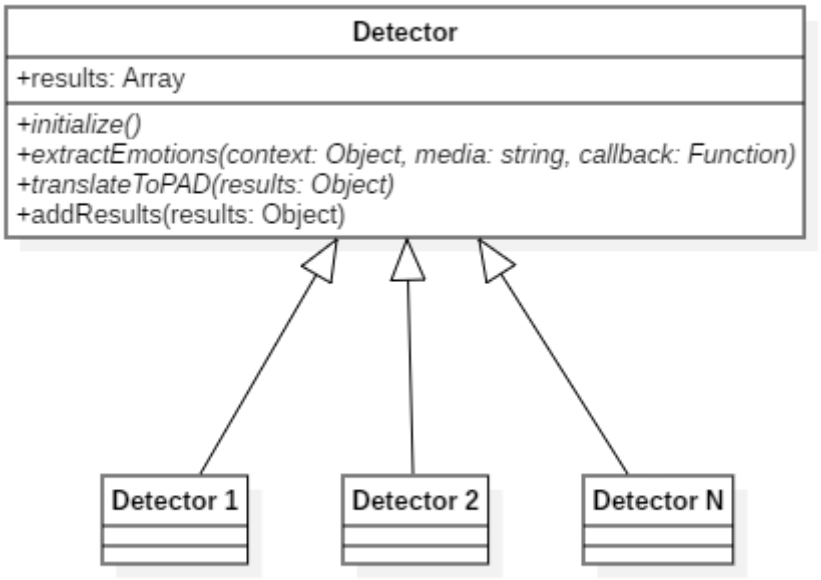

Figura 27. Diagrama de clases de Detector

Diagrama de clases – Integrador de detectores

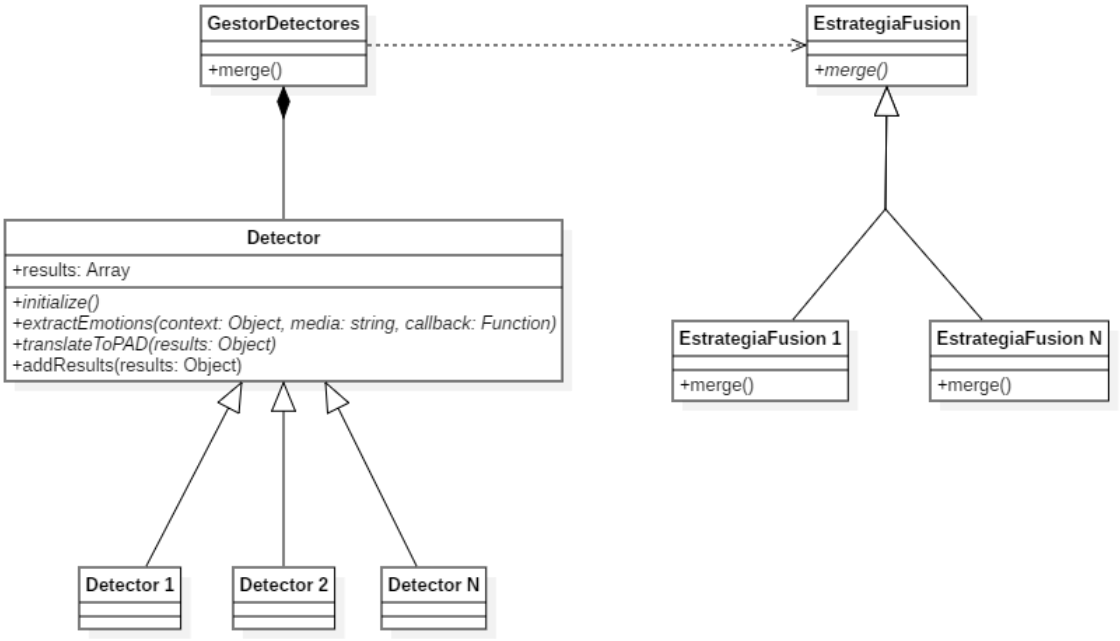

Figura 28. Diagrama de integradores de detectores

Diagrama de componentes del sistema al completo

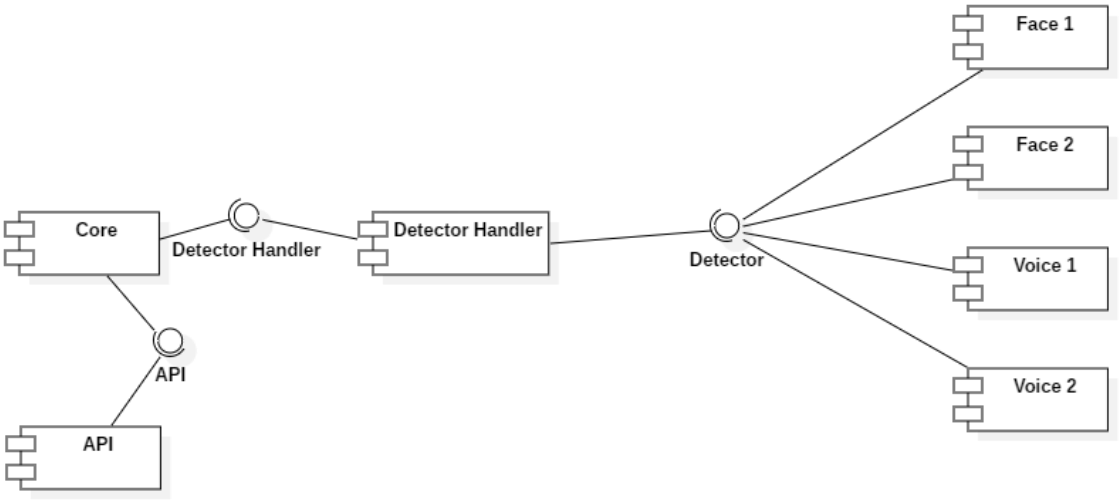

Figura 29. Diagrama de componentes de sistema al completo

Diagrama de secuencia del funcionamiento de un servicio de detección de emociones de terceros

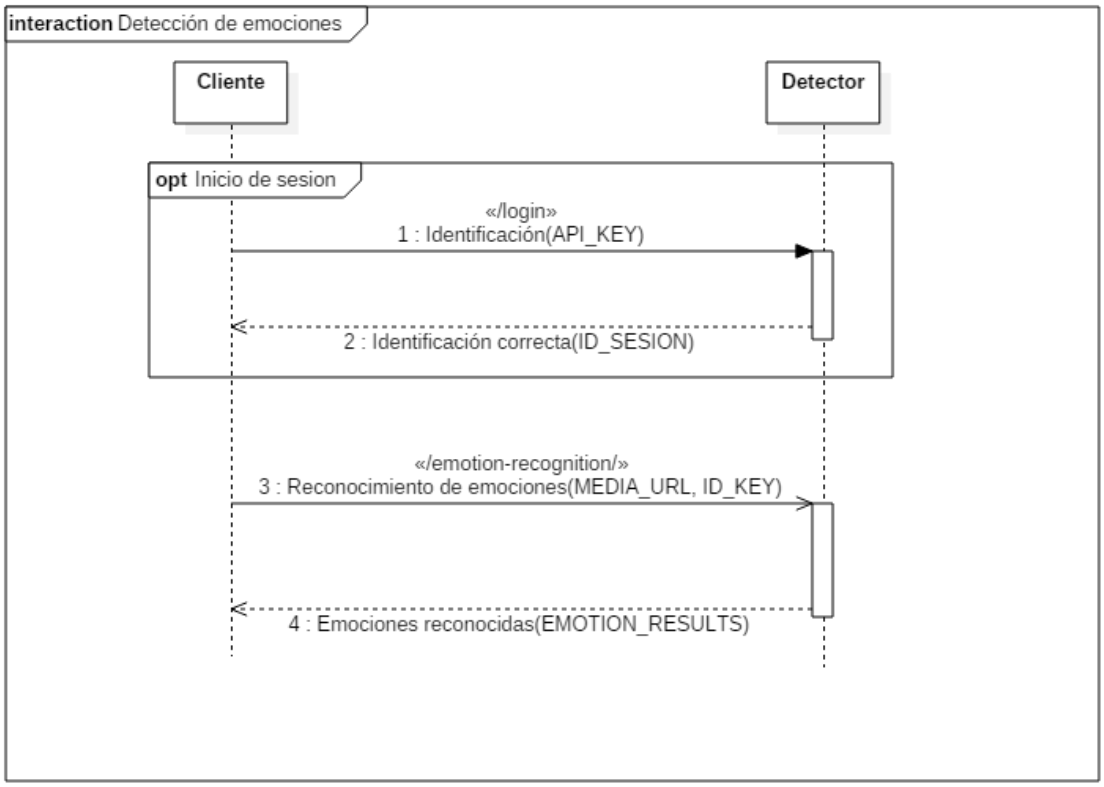

Figura 30. Diagrama de secuencia - Servicio de detección de terceros

Diagrama de secuencia de funcionamiento general de la API

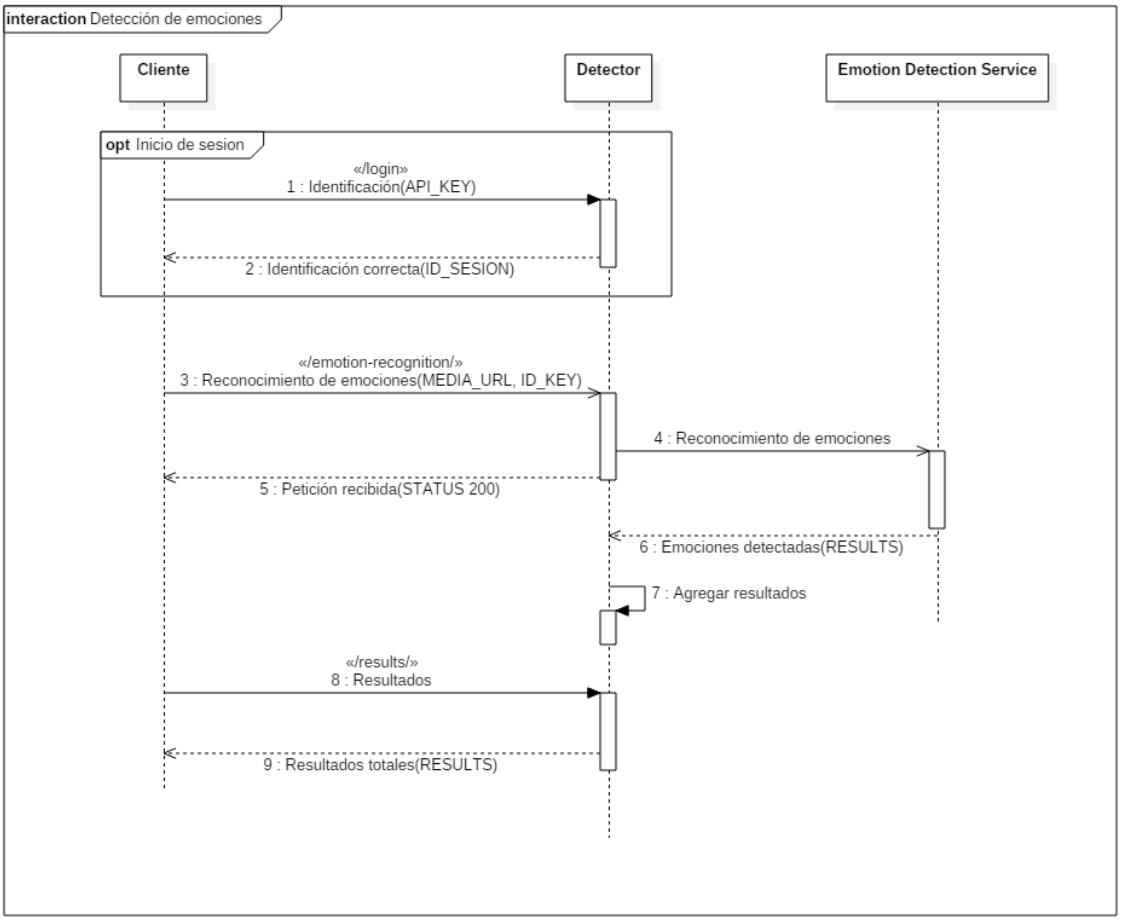

Figura 31. Diagrama de secuencia de funcionamiento general - Sprint 2.

Diagrama de secuencia – Endpoint «/init»

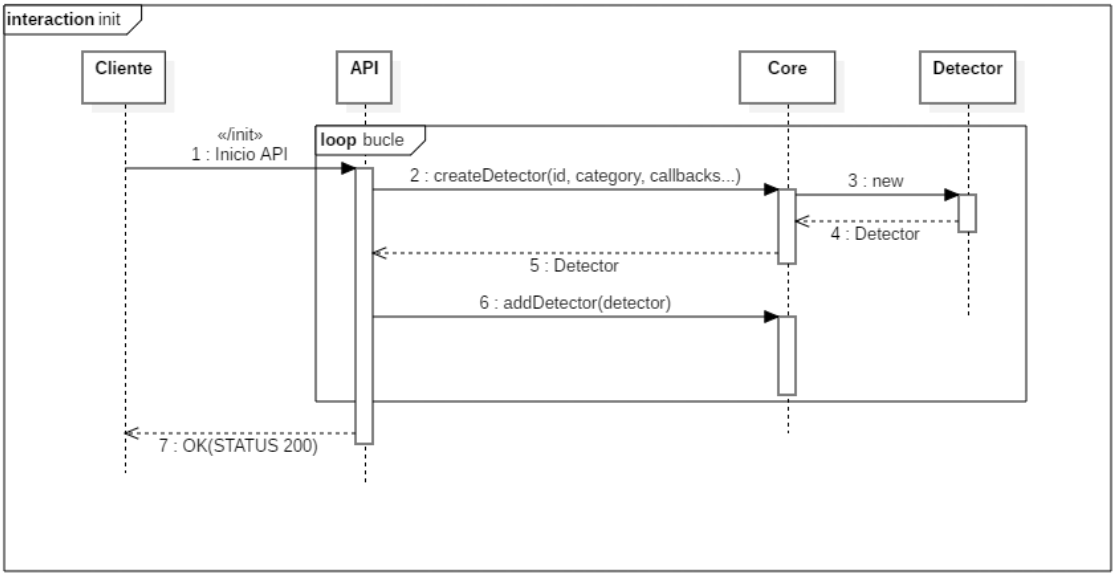

Figura 32. Diagrama de secuencia – Endpoint «/init»

Diagrama de secuencia – Endpoint «/setup»

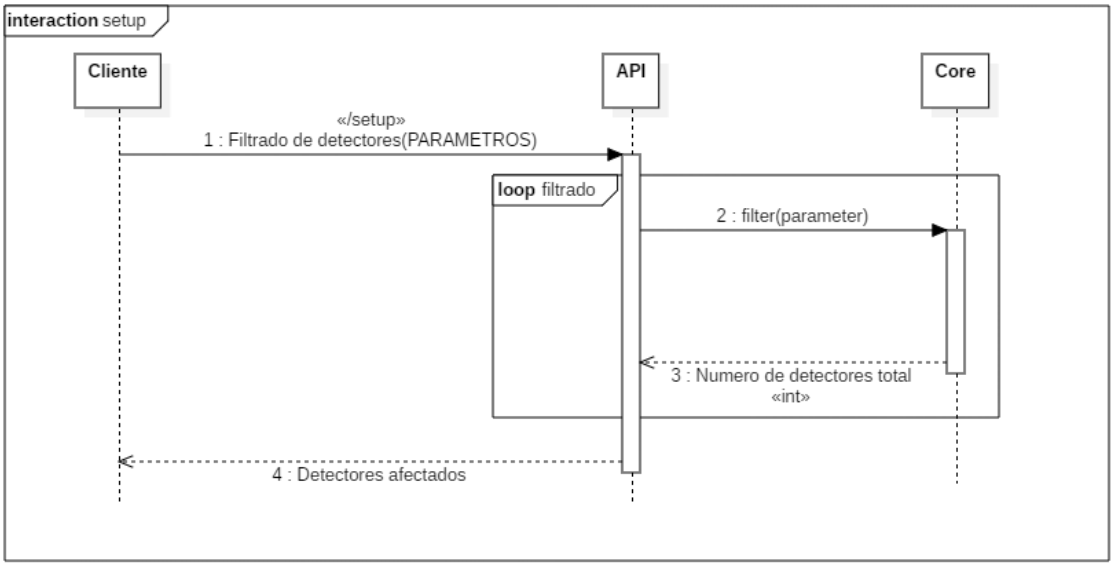

Figura 33. Diagrama de secuencia – Endpoint «/setup»

Diagrama de secuencia – Endpoint «/analyse»

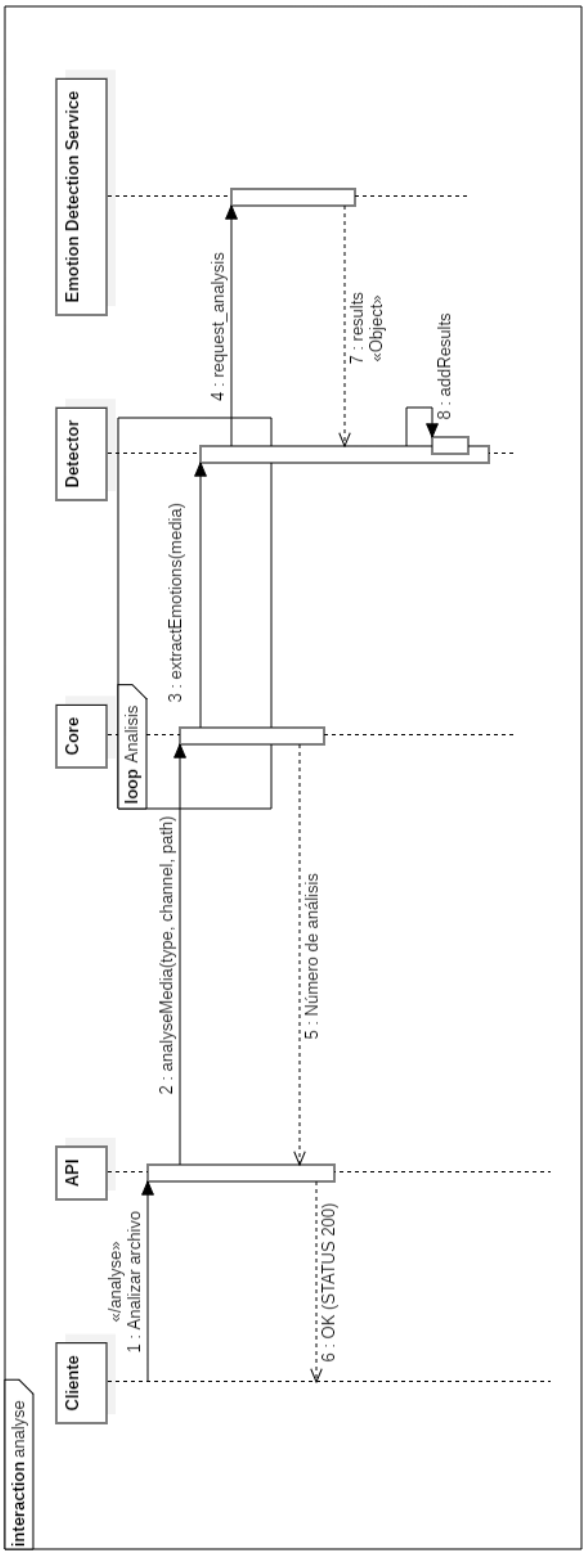

Figura 34. Diagrama de secuencia - Endpoint «/analyse» - Sprint 3.

Diagrama de secuencia – Endpoint «/results»

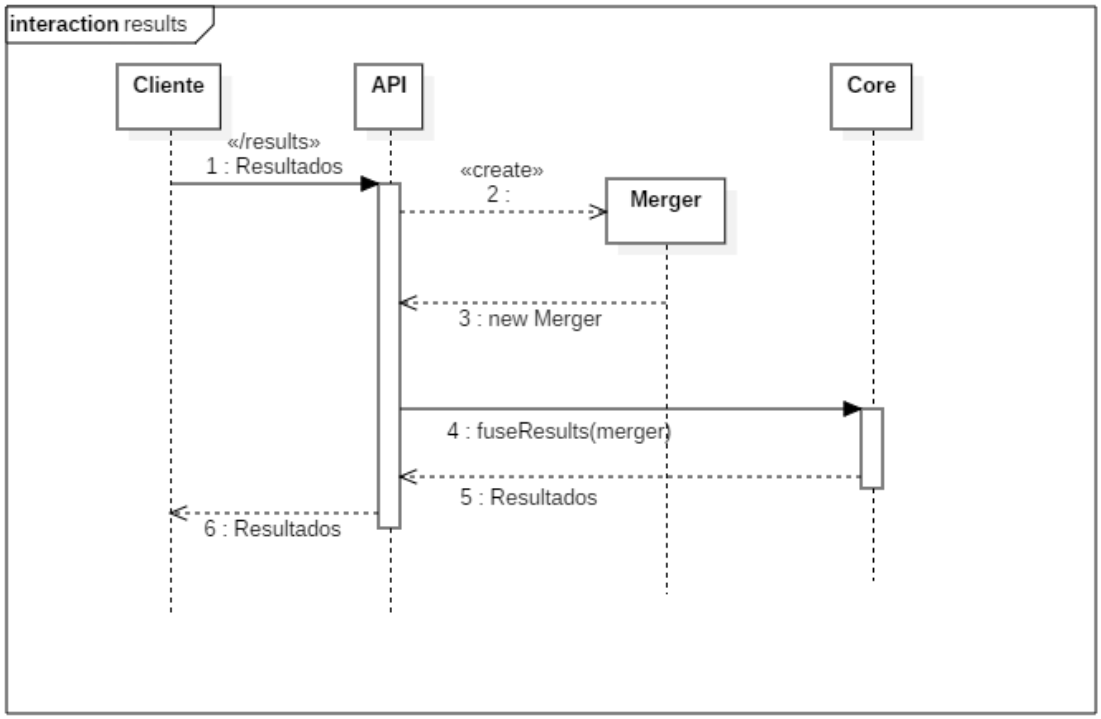

Figura 35. Diagrama de secuencia - Endpoint «/results» - Sprint 3.

Estructura de directorios del sistema

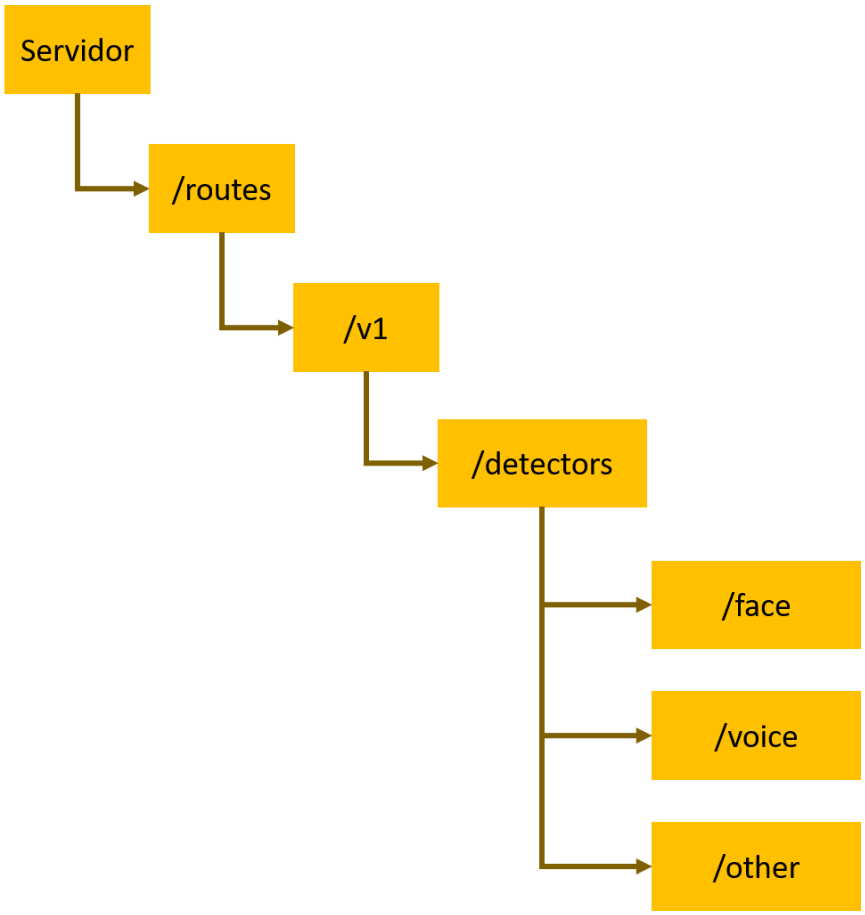

Figura 36. Estructura de directorios del sistema