UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

Computación

TRABAJO FIN DE GRADO

Sistema Multiagente para Rehabilitación Física de Pacientes con Enfermedades Neurológicas

Alejandro Medina Jiménez

Diciembre, 2019

UNIVERSIDAD DE CASTILLA-LA MANCHA
ESCUELA SUPERIOR DE INFORMÁTICA

Departamento de Tecnologías y Sistemas de Información

Computación

TRABAJO FIN DE GRADO

Sistema Multiagente para Rehabilitación Física de Pacientes con Enfermedades Neurológicas

Autor: Alejandro Medina Jiménez

Director: David Vallejo Fernández

Diciembre, 2019

Sistema Multiagente para Rehabilitación Física de Pacientes con Enfermedades Neurológicas

Tribunal:

Presidente: _____

Vocal: _____

Secretario: _____

Fecha de defensa: _____

Calificación: _____

Presidente	Vocal	Secretario
Fdo.:	Fdo.:	Fdo.:

A mis padres ...
... Por aguantar mi caracter incondicionalmente, y creer en mi cuando ni yo mismo lo hacía

Resumen

(... versión del resumen en español ...)

Los accidentes cerebro-vasculares han incrementado durante las últimas décadas. Ocupa el tercer lugar como un factor causante de discapacidad a nivel mundial [39]. Las personas que padecen una enfermedad cerebro-vascular están sometidos a una serie factores que van a deteriorar su calidad de vida.

Debido a este problema, ha aparecido una serie de terapias que han permitido disminuir tanto la mortalidad por accidentes cardio-vasculares, como los problemas que implican a los pacientes padecer estás enfermedades [41]. Las terapias de rehabilitación son un proceso que permite a personas discapacitadas alcanzar mayores niveles de vida, tanto a nivel mental, físico o social.

Como es indicado en el artículo [41], los resultados de recuperación entre los cinco primeros años son mejores cuando la terapía de rehabilitación dura más tiempo. Durante la rehabilitación los pacientes pasan por una fase denominada subaguda en la que uno de los componentes que interviene en el programa de rehabilitación es recuperar la capacidad física. Cuando un paciente pasa por está fase debe maximizar los beneficios en medida de lo posible para recuperar las capacidades perdidas.

Ahora está de moda enfocar la rehabilitación como una serie de actividades que mantengan al paciente entretenido enfocándose en tareas. Este es el punto de la rehabilitación de un paciente en el que interviene el proyecto propuesto. La solución que aporta este proyecto es un sistema que sirva como complemento para impartir la rehabilitación al terapeuta. El terapeuta utilizaría el proyecto durante la terapia, para monitorizar los ejercicios que el paciente debe realizar durante la rehabilitación. Por otro lado, este sistema también permitiría hacer un seguimiento de la evolución del paciente desde casa, enviando toda la información del paciente al terapeuta.

Para ello se propone una interfaz que sea capaz de identificar el movimiento que está realizando el paciente. Esto facilita mucho el uso del sistema al paciente, ya que como se indica en el artículo [41], los pacientes pierden capacidad física y es por ello que es conveniente evitar un exceso de interacción con la aplicación que el paciente utilizaría. Con este fin se propone una interfaz adaptada a una persona en rehabilitación, que se pueda utilizar haciendo uso de simples gestos. A parte de identificar el movimiento del paciente, el sistema sería capaz de evaluar como de bien ha realizado el movimiento de rehabilitación del paciente. Toda está información estaría disponible en el equipo utilizado por el terapeuta para contribuir en la decisión del diagnóstico final.

Para realizar reconocer y evaluar el movimiento del paciente, y gestionar la información que recibe el terapeuta, se propone un sistema multi-agente. Una de las ventajas que aporta el uso de un sistema multi-agente es la escalabilidad, ya que el hecho de usar está arquitectura facilita la integración de nuevos elementos en el sistema. También permitiría atender a un mayor número de usuarios o descentralizar el equipo utilizado para desplegar este sistema.

Abstract

(... english version of the abstract ...)

Strokes have increased over the last few decades. It ranks third as a cause of disability worldwide [39]. People with cerebrovascular disease are subject to a number of factors that will deteriorate their quality of life.

Due to this problem, a series of therapies have appeared that have allowed to diminish the mortality by cardiovascular accidents, as well as the problems that imply to the patients to suffer these diseases [41]. Rehabilitation therapies are a process that allows disabled people to achieve higher standards of living, whether mental, physical or social.

As indicated in the article [41], recovery results within the first five years are better when the rehabilitation therapy lasts longer. During rehabilitation, patients go through a phase called a subacute in which one of the components involved in the rehabilitation program is to regain physical capacity. When a patient goes through this phase, he or she should maximize the benefits as much as possible in order to regain lost capabilities.

It is now fashionable to approach rehabilitation as a series of activities that keep the patient entertained by focusing on tasks. This is the point of a patient's rehabilitation in which the proposed project is involved. The solution provided by this project is a system that serves as a complement to provide rehabilitation to the therapist. The therapist would use the project during therapy, to monitor the exercises that the patient must perform during rehabilitation. On the other hand, this system would also make it possible to monitor the patient's evolution from home, sending all the patient's information to the therapist.

For this purpose, an interface is proposed that is capable of identifying the movement being carried out by the patient. This greatly facilitates the use of the system to the patient, since as indicated in the article [41], patients lose physical capacity and it is therefore advisable to avoid an excess of interaction with the application that the patient would use. To this end, we propose an interface adapted to a person in rehabilitation, which can be used using simple gestures. Apart from identifying the patient's movement, the system would be able to evaluate how well the patient's rehabilitation movement has been performed. All this information would be available on the equipment used by the therapist to assist in the decision of the final diagnosis.

In order to recognize and evaluate the patient's movement, and to manage the information received by the therapist, a multi-agent system is proposed. One of the advantages of using a multi-agent system is scalability, since the fact of using this architecture facilitates the integration of new elements in the system. It would also allow to serve a larger number of users or decentralize the equipment used to deploy this system.

AGRADECIMIENTOS

Es complicado redactar objetivamente está sección sin cometer ningún error. Han sido 4 años que he compartido con un montón de personas, que de un modo u otro han colaborado para formar el ingeniero que pronto seré.

Creo que un buen comienzo es agradecer de nuevo a mis padres que estos años de formación hayan sido posibles. En un principio yo no quería estudiar una carrera, quería hacer un grado superior. En este momento ellos me obligaron a presentarme a selectividad, aprobé y decidí hacer una carrera. A día de hoy considero que es la mejor decisión que he tomado en mi vida, y todo es gracias a ellos. Por no hablar del esfuerzo económico que han tenido que hacer para poder pagarme la carrera.

Tampoco debería olvidarme de mencionar a mi hermano mayor, que me ha orientado y aconsejado durante estos años. A pesar de no haber estudiado la misma carrera que yo, en el momento en el que he tenido algún problema con alguna asignatura que de algún modo toca su formación, me ha ayudado todo lo que ha podido.

A Teresa, mi apoyo incondicional durante este trayecto, una persona capaz de convertir un piso en un hogar. Ha estado ahí tanto cuando las cosas han salido bien, como cuando las cosas han salido mal. Gracias por soportarme y por todo el tiempo que pasamos juntos.

También me en la obligación de mencionar a Ivan, uno de mis compañeros de piso. Me llevo grandes recuerdos de los años que hemos compartido en gran parte de esta carrera. Espero que te vaya genial y que no perdamos el contacto.

Antes de despedirme debo mencionar a David, mi director durante este proyecto. Gracias por tener paciencia conmigo, por aguantar todos los problemas que te he ocasionado y por guiarme para que este proyecto haya finalizado correctamente.

Y por último me gustaría hacer énfasis en que creo que hay mucha más gente que sería digna de mención en está dedicatoria, tanto compañeros como profesores, pero me temo que en una hoja no caben 4 años de carrera.

Alejandro Medina Jiménez

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE LISTADOS

ÍNDICE DE ALGORITMOS

CAPÍTULO 1

INTRODUCCIÓN

Como explica el libro Las nuevas tecnologías y el crecimiento económico en España [35], a lo largo del último siglo, el desarrollo de las nuevas tecnologías informáticas ha aportado cambios en la forma de pensar y de actuar en la sociedad, generando nuevas pautas de comportamiento en el ser humano. Esto ha hecho aumentar la tendencia de las nuevas tecnologías, hasta un punto en el que diversos organismos e instituciones, tales como las administraciones públicas o los centros de salud hacen un uso asiduo de las tecnologías de la información y de la comunicación (TIC).

En un principio, las primeras aplicaciones informáticas sirvieron a aquellos sectores que ejercían funciones de tipo administrativo y financiero, dada la facilidad de resoluciones de problemas numéricos que les permitía manejar en un corto periodo de tiempo.

El uso de aplicaciones informáticas se ha ido extendiendo hasta otros sectores, tales como la industria manufacturera o las cadenas de montaje. Una de las aportaciones que más beneficios ha proporcionado a la sociedad, y a nivel humano, ha sido la introducción de la informática en el campo de la medicina. La informática en el campo de la medicina ha cambiado la forma de abordar los retos médicos, así como muchas otras intervenciones, a las que se debe intervenir a diario.

Una de las ventajas principales que ha supuesto la introducción de las aplicaciones informáticas en el campo de la medicina ha sido el almacenamiento y transmisión de multitud de datos de todo tipo, que aportan valor e información en la consecución del diagnóstico que reciben los pacientes por parte de los expertos médicos.

Esta recopilación de información permite a los profesionales médicos abarcar un amplio abanico de posibles soluciones ante los problemas y enfermedades que puedan padecer sus pacientes. Gracias a la evolución de la tecnología, el tiempo dedicado por el profesional médico a sus pacientes ha sido reducido, facilitando las labores de los terapeutas.

Con el objetivo de realizar una contribución a esta evolución mencionada anteriormente, se pretende desarrollar un sistema de ayuda a distancia. Este sistema será una herramienta para la supervisión de personas que sufren alguna enfermedad neurológica que requiera rehabilitación física y que, por tanto, se encuentran en la necesidad de acudir a su respectivo centro de salud para recibir un tratamiento que se adecúe a sus circunstancias. El objetivo de esta herramienta es mejorar la atención del paciente en fase de rehabilitación, cuando sea atendido por el terapeuta. Esta herramienta podría utilizarla el terapeuta tanto de manera presencial, o como herramienta de telemedicina para hacer más accesible la rehabilitación.

Hay diversas circunstancias familiares y personales, y no todos tienen la misma facilidad para recibir la atención que necesitan, de modo que con ayuda de la aplicación que se pretende desarrollar, serviría de ayuda respecto a la supervisión proporcionada por el clínico a los pacientes, y la calidad del servicio prestado por el profesional; destacando el beneficio en cuanto a las dificultades que podría acarrearle desplazarse al paciente hasta el centro de rehabilitación, dependiendo de su estado.

1.1. REHABILITACIÓN FÍSICA DE PACIENTES

La idea de rehabilitación física esta vinculada al tratamiento que desarrolla una persona para recobrar la condición o el estado que perdió a causa de una enfermedad u otro tipo de trastorno de la salud. De acuerdo con la Organización Mundial de la Salud [45], la rehabilitación busca la restitución de las capacidades de un paciente minusválido. La finalidad es que la persona tenga una vida autónoma, dependiendo en el menor grado posible de los demás.

La rehabilitación física se centra en la funcionalidad corporal. Las tareas de rehabilitación pretenden que el individuo mejore su movilidad y sus habilidades físicas a partir de ejercicios, masajes y otras técnicas. Los ejercicios de rehabilitación suelen realizarse de forma manual y subjetiva [1] cuando el paciente debe practicarlos fuera del centro de rehabilitación, es el profesional encargado de la rehabilitación quien le dice al paciente si lo está haciendo bien o mal de acuerdo a su percepción.

Utilizando una cámara capaz de captar movimientos, usualmente extremidades, y el software adecuado, se puede comprobar si el movimiento de un brazo, por ejemplo, llega al extremo superior y al extremo inferior que se han marcado previamente como puntos clave para ejecutar un determinado movimiento. Esto podría ser una herramienta para complementar el trabajo que realiza el clínico. El sistema almacena información que puede transformarse en conocimiento, ofreciendo datos sobre la evolución de la persona que está llevando a cabo la rehabilitación.

El proceso de rehabilitación es una fase importante tras un accidente cerebrovascular. Además la rehabilitación puede ser un tratamiento lento y frustrante por varias razones, entre ellas que la rehabilitación implica un esfuerzo físico para el paciente.

1.2. MOTIVACIÓN DURANTE LA REHABILITACIÓN

Como se explica en el artículo referenciado [48], el paciente debe superar problemas psicológicos, en los que piensa que no puede realizar los ejercicios que su médico le ha propuesto.

Con el fin de poder ayudar a hacer lo más amenos posibles este tipo de problemas, la herramienta que se desarrolla en este proyecto pretende incentivar al paciente a utilizarla, mostrando como se realizan los ejercicios, mensajes de ayuda, ofreciéndole realizar otros movimientos para motivar al paciente a que tome la iniciativa. El objetivo del diseño del dispositivo sería llamar la atención del paciente con el fin de que realice sus movimientos. Esto serviría de soporte al clínico a la hora de plantear como realizar el proceso de rehabilitación.

Es importante la actitud del paciente con la herramienta ya que este no puede interrumpir el tratamiento, ya el proceso de rehabilitación debe de ser continuo hasta su finalización. Para motivar a los pacientes se llevará a cabo un seguimiento realizado por el clínico donde se podrá ver el estado inicial, los ejercicios a realizar, los últimos avances, el estado de la zona afectada. El paciente al ver una evolución en su recuperación se mantendrá constante con el tratamiento.

En todo proceso de rehabilitación física, tras padecer una enfermedad, es de vital importancia el factor “motivación del paciente”. El paciente experimenta una serie de reacciones ante la llegada de la enfermedad. Son las denominadas fases de adaptación:

- Reacciones de huida o negación: el paciente no quiere aceptar su diagnóstico.
- Reacciones de rechazo: el paciente no quiere enfrentarse al problema ni buscar soluciones.
- Reacciones de racionalización: el afectado se pone en manos de los profesionales y toma conciencia de su proceso rehabilitador.

Es importante el impacto de la atención del terapeuta en la salud del individuo. Esto es debido a que una atención oportuna podría prevenir posibles complicaciones. Además debe prevenir las

reacciones que pueda tener el paciente llegada la enfermedad, y reaccionar ante ellas. Entre las intervenciones podríamos destacar en el contexto de este proyecto la mejora de la movilidad, la prevención de daño osteomuscular o la prevención del dolor realizando un determinado movimiento. Todo esto, está contemplado de algún modo en la herramienta para dar soporte al tratamiento impuesto por el terapeuta.

1.3. IMPACTO SOCIOECONÓMICO

La época de la información ha abierto las puertas a la automatización de maneras inverosímiles. Ha sido posible automatizar tareas que a día de hoy se llevan a cabo en cuestión de segundos. En cambio, anteriormente tenían que ser realizadas por personas y se empleaba mucho más tiempo. Por otro lado hay que tener en cuenta que los ordenadores son mucho más precisos que las personas.

El uso de esta automatización está muy extendido y el número de empresas que dedican personal para realizar tareas monótonas está disminuyendo. En lugar de eso, las empresas están desarrollando sistemas que permitan realizar estas rutinarias tareas de una manera más rápida, económica y eficaz. Estos sistemas deben permitir el trabajo en paralelo, para que solo una parte de la plantilla esté dedicada a supervisar el trabajo realizado por la herramienta. Gracias al éxito de estos sistemas y a los logros en las técnicas de automatización, estos avances están siendo desplegados en zonas que no tienen nada que ver con la industrialización de la fábrica, incluyendo el área de rehabilitación física.

La implantación de este proyecto en una organización, tendrá asociados impactos sociales y económicos. Debido a que con esta herramienta automatizamos parte de la supervisión de ejercicios. Esto contribuye a mejorar la atención recibida por parte de los terapeutas a los pacientes, ya que en vez de tener que supervisar físicamente a todos los pacientes a lo largo de las sesiones, podría segmentar parte de la terapia desde sus hogares. Esto podría hacer más accesible la terapia a los pacientes.

Muchos pacientes, al haber sufrido una lesión de este tipo, no son capaces de desplazarse de manera autónoma al hospital, y por tanto necesitan que alguna persona le lleve, y en caso de vivir en áreas con dificultades de acceso a la ciudad, conducir durante varias horas en algunos casos al hospital, por tanto resulta interesante poder hacer los ejercicios desde casa, evitando de este modo tener que quitar el tiempo de estas personas que trasladan a los pacientes que están recibiendo la terapia.

De otro modo, implantando este sistema en hospitales, podrían servir de soporte a los médicos especializados en este área. Tras su implantación se podría realizar un estudio que verifique si este sistema podría hacer más eficiente la atención a los pacientes que utilicen el sistema.

1.4. PROPUESTA DEL PROYECTO A NIVEL GENERAL

Está propuesta será utilizada como soporte para los médicos cuando impartan terapias a sus pacientes. Este sistema pretende mejorar la experiencia durante la terapia tanto por parte del paciente, como por parte del terapeuta.

Se pretende hacer la terapia más accesible al paciente, y obtener resultados que le proporcionen información respecto a su evolución durante sus ejercicios de rehabilitación. Permitiría aumentar la comunicación entre el paciente y el clínico. Por otra parte, al terapeuta le proporcionaría un entorno en el que recopilar datos del paciente de manera presencial y remota. Además, el sistema es capaz de capturar toda la información procesada en tiempo real recopilada a través de un dispositivo del cuerpo del paciente, y almacenarla. Esto permite al sistema mostrar a modo de apoyo toda la información obtenida del paciente al terapeuta, de este modo podría ver gráficamente como evoluciona el paciente.

También le permitiría contemplar si el paciente está realizando los ejercicios de rehabilitación, ya que es común que los pacientes desistan del tratamiento.

El resultado de este proyecto sería un sistema capaz de reconocer que movimiento está realizando el paciente, reduciendo de este modo la interacción explícita durante el uso del sistema. Para ello sería necesario desplegar un sistema multi-agente donde habría un sistema encargado de reconocer y evaluar movimientos del paciente. Este sistema se encargaría también de almacenar toda la información que va recopilando sobre el paciente. Por otro lado, se haría uso de una cámara tanto por parte del paciente, como por parte del terapeuta para registrar y realizar movimientos de rehabilitación. Finalmente sería necesaria una interfaz desde la que el terapeuta pueda supervisar los ejercicios que realizan los pacientes.

Este proyecto abre varias líneas de trabajos futuros. Se podría definir una línea en la que se capture más información sobre el paciente y que sea procesada por el sistema con diferentes finalidades. Un ejemplo de esto sería añadir funcionalidad para capturar la emoción del paciente tanto durante la realización de un ejercicio, como en líneas temporales durante el proceso de rehabilitación. Por otro lado se podría incorporar un reconocimiento de voz que permita al usuario registrarse, reduciendo de este modo la interacción con el usuario. Incluso se podría contemplar la integración de este sistema como parte de un sistema que se extienda a más áreas de la rehabilitación.

Utilizando la tecnología adecuada se podrían recuperar datos del paciente que puedan ser procesados y permitan al terapeuta obtener conclusiones.

CAPÍTULO 2

OBJETIVOS

Teniendo en cuenta los planteamientos, el contexto y las ideas descritas en la introducción, este apartado está dedicado a qué se pretende llevar a cabo, desarrollando en primer lugar el objetivo general y después detallándolo con objetivos específicos. Consiguiendo así trazar un camino para llegar al resultado final.

2.1. OBJETIVO GENERAL

Con el propósito de implementar un sistema para la rehabilitación física de pacientes con enfermedades neurológicas, se plantea el diseño y desarrollo de un sistema que facilite a las personas que han sufrido un accidente cardiovascular su rehabilitación. Siguiendo este objetivo, el sistema propuesto permitiría dar soporte al terapeuta cuando este usando el sistema con el paciente en la clínica. Otra forma en la que el sistema aportaría valor a los pacientes sería una mejora en la accesibilidad a la rehabilitación. Esto sería posible ya que el sistema facilitaría el acceso a la rehabilitación al paciente desde casa.

El desarrollo de esta propuesta se ha llevado a cabo desplegando un sistema multi-agente compuesto por diferentes agentes, que colaboran con el objetivo de monitorizar la rehabilitación al paciente, y proporcionar los datos recopilados durante el proceso del ejercicio de rehabilitación al terapeuta. Este sistema multi-agente irá conectado con una aplicación que será utilizada por el paciente y otra por el terapeuta. Un requisito de ambas será tener una conexión estable con el servidor. La del paciente estará orientada a realizar ejercicios con la menor interacción posible. Por otro lado la del servidor estará orientada a incorporar nuevas funciones al sistema y monitorizar los datos de uso de cada paciente.

- El paciente, cuando active la herramienta verá los ejercicios que tiene que realizar para ir completando el tratamiento, a medida que los vaya ejecutando, el sistema le proporcionará información acerca de la exactitud de sus movimientos con respecto a lo estipulado por el clínico.
- El clínico, podrá almacenar los diferentes movimientos que quiere que ejecute el paciente, en función del tratamiento que necesite determinará los diferentes movimientos a los distintos pacientes, además de una vez que el paciente ha finalizado sus ejercicios poder ver con que exactitud los ha realizado, además el sistema le recomendará nuevos movimientos asociados a la enfermedad especifica del paciente.

2.2. OBJETIVOS ESPECÍFICOS

A continuación se presentarán los objetivos específicos que se pretenden abordar, derivados del objetivo general. Estos pueden agruparse en:

- Reducir la interacción con los pacientes gracias al clasificador automático, evitando la interacción explícita, que suele dar lugar a confusión y errores.
- Facilidad de uso, utilizando un diseño simple que un usuario inexperto pueda aprender sin problemas. Uno de los propósitos que se pretendía satisfacer era el diseño de una interfaz sencilla e intuitiva, para que cualquier persona pueda utilizarla sin problemas. Esto se ha conseguido permitiendo a los usuarios ver las imágenes generadas por la cámara, permitiendo también que la mayoría de las opciones se activen por comandos de voz para que los pacientes con movilidad reducida no tuviesen que depender del teclado y del ratón u otros periféricos.
- Motivación a seguir utilizando la herramienta, para evitar que el paciente deje de utilizarla durante el tratamiento. Es importante la actitud del paciente con la herramienta ya que este no puede interrumpir el tratamiento, ya el proceso de rehabilitación debe de ser continuo hasta su finalización. Para motivar a los pacientes se llevará a cabo un seguimiento realizado por el clínico donde se podrá ver el estado inicial, los ejercicios a realizar, los últimos avances, el estado de la zona afectada. El paciente al ver una evolución en su recuperación se mantendrá constante con el tratamiento.
- Escalabilidad, permitiendo así añadir más agentes por si en algún futuro fuese necesario añadir más funcionalidad o ampliar el sistema.

CAPÍTULO 3

ESTADO DEL ARTE

3.1. INTELIGENCIA ARTIFICIAL DISTRIBUIDA

Haciendo referencia el artículo **Inteligencia Artificial Distribuida y Razonamiento Basado en Casos en el Conocimiento** [28], se define la inteligencia artificial distribuida como un subconjunto de la inteligencia artificial centrado en comportamientos inteligentes compuestos por procesos independientes. Es común asociar un sistema multi-agente a la inteligencia artificial distribuida, ya que se podría decir que los agentes son esos procesos que cooperan para obtener ese comportamiento inteligente.

A continuación, se desarrollará el concepto de agente inteligente, para después explicar como trabajan los agentes dentro de un entorno multi-agente. Después se hablará sobre plataformas que permitan el desarrollo de sistemas multi-agentes, y para finalizar este apartado aplicaciones del uso de un sistema multi-agente en el campo de la medicina.

3.1.1. El concepto de agente inteligente

Referenciando el libro de **Agentes inteligentes: definición y tipología. Los agentes de la información.** [24]. Un agente inteligente es una entidad capaz de recopilar información del entorno que le rodea, procesarla y actuar de un modo u otro en base a la información que ha procesado.

Cuando procesa está información realiza una serie de operaciones que permiten satisfacer sus necesidades, en caso de que sea posible alcanzarlas. Todos los agentes inteligentes son programas, y son considerados entidades individuales. Están constantemente a la escucha de algún evento que les indique como actuar en una situación determinada.

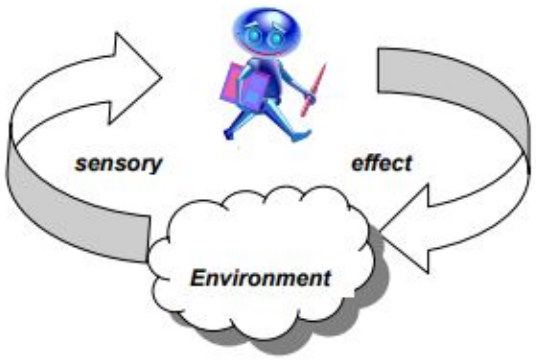

Figura 3.1: Concepto de agente inteligente
https://aisel.aisnet.org/cgi/viewcontent.cgi?article=3273&context=cais

Los agentes inteligentes desde el punto de vista de la inteligencia artificial poseen una serie de características:

- Autonomía: los agentes están dotados de la capacidad de actuar sin requerir ningún tipo de intervención explícita por parte de un usuario.
- Sociabilidad: hace referencia a la capacidad de interactuar con otros agentes. Para ello deben utilizar un lenguaje que ambos agentes conozcan. Los agentes también podrían interactuar con humanos por diferentes medios.
- Capacidad de reacción: los agentes son capaces de percibir el entorno por medio de sensores, y son capaces de actuar en función del estado del entorno que perciben, por medio de actuadores.
- Iniciativa: Los agentes pueden actuar por iniciativa propia para resolver un problema que se plantee en una situación determinada.

Para que un agente pueda realizar acciones, requiere de un entorno que pueda cambiar y percibir. Los entornos de agentes se pueden clasificar en función del grado en el que afectan a las decisiones del agente, como se explica en [50]. Para clasificar los entornos, habría que realizar todas las combinaciones de las diferentes características que puede tener un entorno. Estás características son:

- Accesible vs inaccesible: un entorno accesible es aquel del que un agente puede extraer una gran cantidad de información. Por el contrario, un entorno inaccesible es aquel del que se puede extraer poca información.
- Determinista vs no determinista: una acción en un entorno determinista garantiza un único resultado, por tanto en un entorno no determinista habría incertidumbre tras realizar una acción.
- Episódico vs secuencial: un entorno episódico es aquel en el que hay varios escenarios y no guardan relación entre sí.
- Estático vs dinámico: un entorno estático permanece en el mismo estado en el que el agente llegó a él, mientras que un entorno dinámico permanece en constante cambio.
- Discreto vs continuo: se dice que un entorno es discreto si el número de percepciones distintas es finito.

3.1.2. Sistema multiagente

En el artículo[28] se da una definición para una sociedad de agentes. Una sociedad de agentes es una red interconectada en la que cada nodo de la red representa a un agente independiente. Es habitual que se defina un lenguaje de comunicación para intercambiar mensajes entre los diferentes agentes de la red. Los agentes no tienen porque intercambiar información de manera directa entre ellos, sino que podrían utilizar para ello un intermediario. Basándose en este último comentario se podrían clasificar las sociedades de agentes en dos, las centralizadas y las descentralizadas.

Los sistemas multi-agentes son similares a los sistemas distribuidos, pero a diferencia de ellos están dentro del marco de la inteligencia artificial [38]. A la hora de diseñar un sistema multi-agente se pueden dar varios enfoques:

- **Enfoque clásico:** trata de aumentar la inteligencia de los agentes para que el sistema en términos generales se beneficie de este incremento de inteligencia particular de cada agente. Se suelen dar definiciones formales mediante un sistema formal para especificar el problema que la comunidad de agentes pretende resolver.

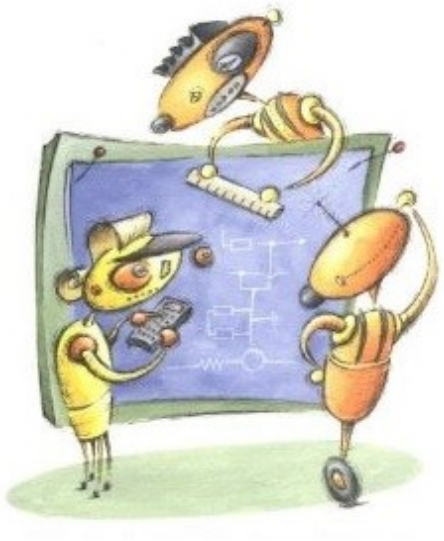

Figura 3.2: Sistema multi-agente
https://www.ecured.cu/images/5/54/Sistema_multiagentes.jpg

- **Enfoque constructivista:** en vez de intentar de potencial la inteligencia de cada agente para que se beneficie el sistema de las mejoras individuales de cada uno, se trata de aumentar la inteligencia del sistema, independientemente de la inteligencia incorporada en cada agente.

A continuación se indicarán las características más relevantes que debe tener un sistema multi-agente [12]:

- Autonomía: los agentes deben ser capaces de operar sin requerir de ningún tipo de intervención humana o de otros agentes.
- Sociabilidad: los agentes deben poder interactuar con otros agentes o con personas.
- Reactividad: pueden percibir eventos dentro de su entorno y poder emitir una reacción.
- Iniciativa propia: a pesar de que no haya cambios en el entorno, los agentes deben ser capaces de iniciar una acción para satisfacer sus propios objetivos.
- Veracidad: los agentes no deben enviar información errónea intencionadamente.
- Cooperación: cada agente debe estar dispuesto a ayudar a otros agentes dentro de la comunidad.
- Racionalidad: un agente actuará siempre en beneficio de sus objetivos siempre y cuando estos objetivos se puedan cumplir.

Basándose en la forma de tomar las decisiones de cada comunidad de agentes, se pueden definir diferentes modelos de arquitectura en función de como los agentes se organicen a la hora de recopilar una percepción y reaccionar a través de una acción del sistema en conjunto.

- Arquitectura deliberativa: pueden ser horizontales y verticales. Respecto a las primeras, cada agente recibe su propio modelo de percepción y proporciona una acción que posteriormente será contrastada con la acción del resto de los agentes. Por otro lado, si es vertical se pueden dar dos casos. En el primero un agente obtiene una percepción que va siendo manipulada por el resto de agentes hasta generar una acción, y en el segundo la percepción pasa por todos

los agentes, pero al llegar al último vuelve al primero, y en este transcurso todos los agentes vuelven a manipular la acción en relación con las indicaciones del último, siendo el agente que tomo la percepción el que ejecuta la acción.

- Arquitectura reactiva: la forma de obtener la percepción es similar a una arquitectura deliberativa horizontal, ya que todos los agentes obtienen la percepción del entorno. Después, un agente genera la acción en relación con sus propios intereses, está acción va pasando por todos los agentes, los cuales la manipulan, hasta que finalmente llega al último agente que envía la acción resultante manipulada en función de los intereses de cada agente del sistema.
- Arquitecturas híbridas: combina las características de cada una de las arquitecturas mencionadas previamente.

Los sistemas multi-agentes son aplicados en el mundo real en multitud de aplicaciones, destacando redes y móviles, juegos de ordenador, películas, sistemas de defensa coordinados. También se puede incluir logística, sistemas de información geográfica, gráficos, diagnósticos médicos, entre otros campos.

3.1.3. Plataformas de desarrollo

A continuación se van a exponer diferentes alternativas que se han estudiado durante este proyecto para escoger la plataforma donde se ejecutarán los diferentes agentes que componen el sistema. Se analizarán las características de cada plataforma y las ventajas que aportan:

- ZeroC ICE: Este ha sido el middleware con el que se ha implementado este proyecto, se ha decidido utilizar por muchos motivos que hacen de él una plataforma más versátil que sus competidores. ZeroC Ice se puede interpretar como un conjunto de herramientas orientado al desarrollo de objetos distribuidos. Se utiliza durante el desarrollo de sistemas distribuidos, y está preparado para hacer transparentes las características de la comunicación entre los nodos del sistema al programador. Entre las características de la plataforma podemos destacar:
  - Provee la plataforma adecuada para ser usada en entornos heterogéneos ya que soporta una gran cantidad de lenguajes de programación diferentes y permite la comunicación entre ellos, por tanto, podrían haber sido diseñados cada agente del sistema en un lenguaje de programación distinto y haber mantenido la comunicación entre ellos sin ningún problema.
  - Proveer de las características necesarias para permitir desarrollar aplicaciones distribuidas realistas en una amplia gama de dominios.
  - Evitar una complejidad innecesaria, haciendo la plataforma fácil de usar y entender. Gracias a esto el programador puede centrarse más en la implementación del comportamiento de los agentes, haciendo transparente la implementación de las comunicaciones, dejándolo todo en manos de este middleware.
  - Hace una gestión muy eficiente del ancho de banda, uso de memoria y de CPU en tiempo de ejecución.
  - Integra en el sistema la implementación de mecanismos de seguridad, haciéndolo adecuado para un entorno en producción, usado sobre redes públicas.

  ZeroC ICE describe un objeto como una abstracción caracterizada por ser una entidad local o un espacio remoto de direcciones capaz de responder peticiones de clientes, un objeto de este tipo puede ser instanciado en uno o en múltiples servidores. Además cada objeto de ZeroC Ice tiene una o más interfaces, este es un punto importante, ya que una interfaz es una

colección de las operaciones con nombres que son ofrecidas por el objeto, y los clientes realizan peticiones invocando a estas peticiones. Éstas, al igual que un método convencional, pueden tener, o no, parámetros y valores de retorno, en caso de tener parámetros, los valores de estos son pasados por el cliente, y en el caso de tener valor de retorno, este valor es devuelto por el servidor al cliente.

- JACK: Como se indica en el artículo [11], JACK provee de un entorno de desarrollo orientado a agentes, está construido sobre Java y completamente integrado con este lenguaje de programación. Incluye todas las componentes del entorno de desarrollo de Java, y adicionalmente, incluye extensiones para dar soporte al diseño y ejecución de agentes:

  - Estructuras para comunidades de agentes.
  - Control en tiempo real.
  - Sistemas de comunicación entre agentes.

  La plataforma JACK cuenta con una interfaz gráfica amigable que hace de su utilización algo más sencillo. Requiere de una gran cantidad de recursos en relación con otras plataformas durante la ejecución y provee todo lo necesario para la ejecución de los agentes. Permite generar código de agentes y editarlo para adaptarlo a la solución que se proponga. Utiliza un analizador para chequear que el código agregado a los agentes o a archivos externos a los agentes sea correcto, y posee un entorno de ejecución que permite visualizar de una forma sencilla los agentes desarrollados.

- ZEUS: Es una herramienta que provee un entorno integrado para el desarrollo de sistemas multi-agente distribuidos como se indica en [43]. El uso de esta herramienta facilita el desarrollo de agentes que interactúan entre ellos, haciendo más ágil el desarrollo. Esto es posible gracias a las técnicas propias de la plataforma como por ejemplo, la reutilización de código o la generalización de las tecnologías de cada agente. ZEUS está orientado a un desarrollo rápido, altamente personalizable y escalable. Es habitual cuando se trabaja con esta plataforma acotar el alcance del proyecto para especificar las capacidades que debe tener el resultado.

  Por otro lado, la herramienta permite el uso de un diseño que permita extenderlo fácilmente y configurar el número de agentes que va a tener el sistema, la organización y la forma en la que van a interactuar. Los agentes en esta plataforma pueden tener diferentes roles en función de la autoridad que tienen sobre otros agentes:

  - Superior: Tiene más autoridad que el agente de referencia.
  - Compañero: Tiene la misma autoridad que el agente tomado por referencia.
  - Subordinado: Tiene menos autoridad que el agente de referencia.

- JADE: Es un marco para el desarrollo de aplicaciones multi-agente haciendo referencia a [3]. Está desarrollado bajo las especificaciones del estándar FIPA, y ofrece una librería para la implementación de la comunicación y la arquitectura de los agentes. FIPA es un estándar que especifica las normas de convivencia, funcionamiento y gestión de una sociedad de agentes. Jade solo soporta el lenguaje de programación Java para el desarrollo de los agentes. Por otro lado, permite operaciones sobre la comunidad de agentes como la movilidad de los agentes y reutilización de un una amplia gama de objetos ya predefinidos.

- MadKit[23]: Es otra plataforma desarrollada por LIRMM (Laboratorie dÍnformatique de Robotique et de Microélectronique de Montpellier), a diferencia de JADE actualmente no cumple el estándar FIPA. Esta implementada en lenguaje Java con la finalidad de poder ejecutarla en diferentes entornos. Una de las principales características de la plataforma MadKit y que la diferencia claramente del resto de plataformas es la utilización de un modelo de organización

basado en los conceptos de Agente, Grupo y Rol. Con este modelo se consigue una organización consistente de los elementos y relaciones que surgen dentro de los sistemas multi-agentes. Madkit forma una capa intermedia entre la máquina virtual de Java y los sistemas multi-agente, proporcionando un entorno en donde los agentes se pueden ejecutar y comunicar entre sí.

3.1.4. Aplicaciones en medicina

Los agentes y los sistemas multi-agentes son cada vez más relevantes en el desarrollo de sistemas dinámicos y distribuidos, adquiriendo una gran aceptación en áreas en las que son implantados. En este apartado se va a hablar sobre las aplicaciones de los sistemas multi-agente en el campo de la medicina.

Aunque hay mucha diversidad de problemas cuando se trata del cuidado de la salud, la mayoría de los problemas tienen partes comunes a la hora de tratarlos en un centro médico. Cuando el paciente está en un centro médico, no se realizan todas las pruebas en la misma sala. Tampoco le atiende constantemente el mismo médico, sino que va pasando por diferentes especialistas, cada uno especializado en un determinado área [40]. Un sistema multi-agente en un entorno como es el de los centros de salud, es una buena opción ya que en estos espacios la actividad no está centralizada en un único punto, si no que está distribuida en diferentes lugares.

A continuación se expondrán una serie de casos reales de sistemas multi-agentes implantados en el campo de la medicina:

- **eMAGS**[46] es un sistema orientado a la transmisión de información médica en un entorno con una alta tasa de transmisión de información. Este sistema se ha diseñado centrándose en los entornos generados por los sistemas médicos en el campo de la ontología y en una norma para mensajes sanitarios denominada Nivel de Salud 7 (HL7). El marco sobre el que funciona el sistema, es un entorno multi-agente que proporciona un entorno para la interacción de los sistemas médicos basándose en un enfoque cliente-servidor.

- **ALZ-MAS**[53] está basado en un sistema multi-agente orientado a mejorar los cuidados sanitarios a los pacientes que padecen Alzheimer. El sistema está constantemente recopilando información automáticamente de los pacientes y del entorno desde diferentes nodos del sistema con el objetivo de mejorar la asistencia a los pacientes. Su distribución es homogénea y se centra en la inteligencia ambiental.

- **GerMAS**[47] al igual que el sistema mencionado anteriormente está orientado a residencias geriátricas. A diferencia del caso interior, este sistema personaliza la experiencia de cada paciente haciendo uso de la tecnología NFC y RFID para por ejemplo, identificar las medicinas que toma cada paciente. El sistema multi-agente está dotado de la capacidad de razonamiento para analizar la situación de cada paciente, y mejorar su atención sanitaria en base al análisis realizado.

- **Sistema multi-agente difuso-probabilístico para la evaluación del riesgo de cáncer de mama**[54]. Este sistema utiliza técnicas de lógica difusa y algoritmos probabilistas para determinar el riesgo de desarrollo de cáncer de mama en las mujeres. El sistema está compuesto por varios agentes difusos que aplican la teoría de conjuntos difusos, para evaluar el riesgo de padecer cáncer. Esto se lleva a cabo mediante la evaluación de probabilidades difusas de alto o bajo riesgo. El resultado de estas evaluaciones, contribuye a que el sistema multi-agente toma decisiones para determinar el nivel de riesgo del paciente de padecer está enfermedad.

3.2. SERIES TEMPORALES

Una serie temporal es una secuencia de información que representa observaciones o valores, medidos en determinados instantes de tiempo y ordenados cronológicamente. Los datos pueden estar espaciados a intervalos.

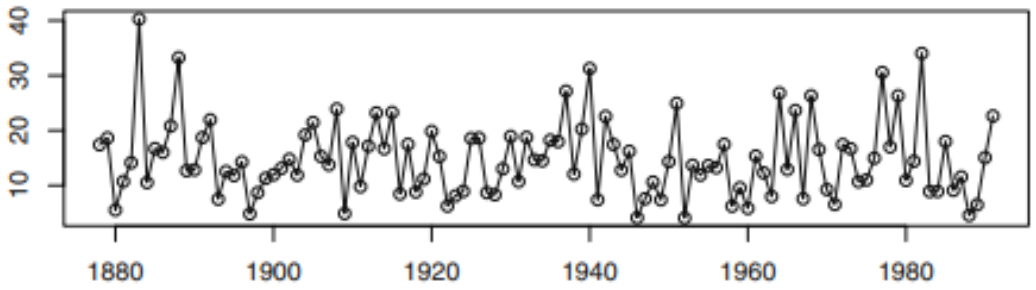

Figura 3.3: Ejemplo de serie temporal
https://link.springer.com/chapter/10.1007/978-0-387-75959-3_1

3.2.1. Introducción

Hoy en día no es muy complicado encontrar series temporales, ya que están presentes en un amplio abanico de sectores [9]. Ejemplos de esto podrían ser la evolución del precio de un producto, los beneficios de una empresa, la tasa de mortalidad por año, la evolución del número de habitantes de un país, la evolución de la temperatura en un área determinada, la medición de residuos en un flujo liquido o la tensión de un paciente.

Por tanto se puede apreciar que la mayoría de la población utiliza diariamente series temporales sin ser realmente conscientes de ello. Básicamente una serie temporal es una secuencia de observaciones ordenadas cronológicamente en el tiempo sobre una característica, que como se ha visto arriba, varía en función del lugar de donde la serie ha sido extraída. También es posible evaluar varias características en vez de una, en cuyo caso se hablaría de series vectoriales o multivariante.

Tras esta breve definición se podría interpretar una serie temporal como un vector en el que cada elemento del vector representa una instancia en el tiempo de la serie a analizar, y el contenido de cada elemento representa el valor que ha tomado la característica que está siendo analizada en ese determinado instante. Para poder recoger la serie temporal en un vector sería necesario que la longitud del vector sea igual al número de elementos de la serie.

A continuación se va a proceder a describir los posibles objetivos que pueden tener analizar series temporales [37]. Analizar series temporales consiste en elaborar un modelo que describa de forma adecuada como influyen los elementos que forman parte de la serie a lo largo del tiempo. Este modelo permitirá a los analistas describir gráficamente como ha evolucionado la serie temporal, pudiendo buscar una relación entre los valores que se muestran gráficamente y la causa de porque ese instante de tiempo ha tomado este valor. Del mismo modo, con esta información la persona encargada de analizar la serie podría buscar en base a los resultados obtenidos, una forma de prevenir el valor en futuros elementos temporales de la serie.

3.2.2. Métodos de comparación

Haciendo referencia al artículo [58]. Las series temporales aportan valor a la hora de realizar un análisis respecto al motivo por el que se han producido los valores de cada elemento, pero en determinadas situaciones es conveniente poder comparar dos series completas. El problema es que

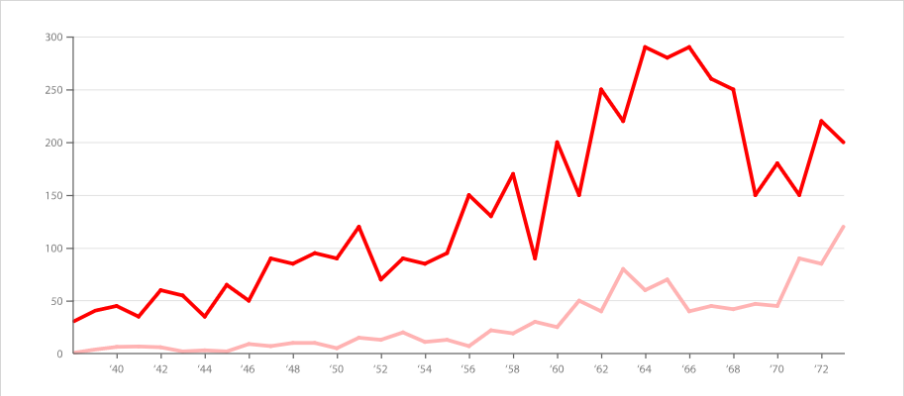

Figura 3.4: Comparación de series temporales
https://datavizcatalogue.com/ES/metodos/grafica_de_linea.html

no es fácil comparar dos series temporales debido a que es habitual que las series sean especialmente largas, y esto hace que trabajar directamente con ellas puede llegar a ser computacionalmente caro. En este momento surge la necesidad de averiguar diferentes métodos de comparación, utilizando finalmente el método que mejor se adapte a las necesidades de un problema específico.

A continuación se listarán los métodos de comparación más relevantes junto a las características que aportan respecto a otros métodos de comparación.

- **Discrete Wavelet Transformation** [22] hace referencia al método de análisis para series temporales. Está formado por cuatro componentes básicos implícitos en la serie temporal:
  - Tendencia: describe la evolución de la serie a lo largo del tiempo.
  - Variación estacional: repetición de un patrón en la serie a lo largo del tiempo.
  - Variación cíclica: similar a la variación estacional salvo por que el patrón cada vez tarda más, o menos tiempo en reproducirse.
  - Variación aleatoria: también denominada residuo, no muestra ninguna regularidad y se obtiene una vez eliminadas la tendencia y las variaciones cíclicas de la serie.

  Una de las ventajas que supone utilizar este método de comparación es que las dos series a comparar pueden tener una longitud diferente. Para realizar la comparación es necesario tomar una serie de referencia, y el valor resultante de la comparación indicará como se similar es la serie que se prueba, respecto a la serie tomada por referencia. En pocas palabras este algoritmo realiza la comparación de dos series estirando o encogiendo la serie de prueba, hasta que la longitud de esta coincida con la serie tomada por referencia. Después compara el punto de cada serie equivalente con la serie opuesta y en base a estas comparaciones genera un resultado.

- **Discrete Fourier Transformation**[15] es un método de comparación que logra mejorar el rendimiento de algoritmos clásicos exponencialmente. Es utilizado principalmente en teoría cuántica ya que es capaz de resolver problemas con una complejidad exponencial en ordenadores cuánticos en un tiempo polinómico.

- **Single Value Decomposition**[57] es un algoritmo utilizado principalmente en problemas genéticos. Las series temporales genéticas son generalmente ruidosas. Este algoritmo permite obtener pequeñas señales de datos con un valor alto de ruido.

- **Discrete Cosine Transformation**[52] al igual que el algoritmo Discrete Fourier Transformation, este algoritmo trabaja con series temporales finitas. Es un algoritmo presente en

varios formatos de imágenes (JPEG el más relevante) ya que es caracterizado por ser capaz de concentrar una gran cantidad de información en poco espacio. Al utilizarlo en imágenes reduce la cantidad de errores al comprimirlas.

- **Piecewise Aggregate Approximation** [60] utiliza su propia forma de representación. Utilizar este algoritmo a la hora de comparar dos series puede reducir los tiempos de cómputo. Esto lo logra realizando una aproximación desde el límite inferior de la serie por partes. La desventaja de utilizar este algoritmo es la perdida de precisión producida durante la aproximación.

- **Chebyshev polynomials** [36] se recurre a ellos en una gran cantidad de casos y actualmente ocupan una posición relevante en desarrollos de la actualidad. Haciendo uso de reglas para definir polinomios, este algoritmo permite realizar una aproximación de la distancia entre dos series temporales.

3.2.3. Evaluación de ejercicios de rehabilitación

El proceso de rehabilitación permite a los pacientes ir eliminando progresivamente las discapacidades o las deficiencias, que sufren tras padecer una enfermedad o accidente [14]. Entre las actividades que debe realizar un paciente durante este proceso se incluye la realización de ejercicios físicos.

La mayoría de los médicos están de acuerdo en buscar un estándar, a la hora de evaluar la forma de realizar los ejercicios de rehabilitación en pacientes [10]. Por tanto es necesario encontrar una métrica que permita definir como debe hacer un paciente un ejercicio para que este sea correcto, y en que factores debe apoyarse el terapeuta para decidir que el ejercicio se ha realizado de forma incorrecta.

Figura 3.5: Evaluación de ejercicios de rehabilitación
https://www.dolor.com/fisioterapia.html

Para evaluar los ejercicios de rehabilitación, es necesario medir el rendimiento del ejercicio y garantizar que los pacientes realicen correctamente los ejercicios, cuando lleguen a su casa [8]. Lo más común es que los ejercicios de rehabilitación se hagan en un entorno hospitalario. El problema es que con el paso del tiempo, aumenta la necesidad de tener un servicio sanitario más eficiente. Una de las formas de ser más eficiente a la hora de realizar ejercicios de rehabilitación es haciendo una porción del total de los ejercicios mandados por el clínico en casa. El problema es que muchos pacientes encuentran dificultades al realizar sus ejercicios de rehabilitación en su hogar. Uno de los problemas más importantes es la falta de la supervisión de un terapeuta cualificado para realizar está labor, ya que los pacientes podrían realizar los ejercicios de una manera incorrecta.

Si un paciente realiza sus ejercicios de rehabilitación de una forma inadecuada podría repercutir gravemente en el proceso de rehabilitación. A la hora de evaluar si un ejercicio se está realizando de la forma correcta, hay que tener en cuenta una serie de métricas:

- La alineación correcta de las articulaciones durante el ejercicio.
- Realizar el ejercicio con una velocidad de movimiento adecuada, ya que tanto el exceso como el defecto de velocidad, respecto a la velocidad indicada por el terapeuta, podrían tener un efecto adverso en la rehabilitación.
- La mala calidad del ejercicio, ya un terapeuta es capaz de corregir a un paciente en la postura en la cual realiza el ejercicio, pero esto no es posible si el paciente está en su casa.

Actualmente muchos centros de investigación trabajan en soluciones que permitan la monitorizar ejercicios de rehabilitación de manera remota, permitiendo que el paciente obtenga resultados tras la realización de sus ejercicios y pueda saber si los realiza adecuadamente. Muchas de estas soluciones utilizan series temporales a la hora de evaluar las métricas mencionadas anteriormente.

El algoritmo de deformación dinámica del tiempo (DTW) permite la evaluación y reconocimiento de ejercicios motrices [56]. Esto se debe a que permite que la serie que se procede a evaluar sea incompleta en la variante del algoritmo de inicio y final abierto. Gracias a esta variante, el algoritmo es capaz de intentar buscar una relación entre la serie de referencia y la serie a evaluar, incluso si la serie que se evalúa es el sufijo o prefijo de la serie tomada por referencia.

Haciendo referencia a los resultados obtenidos en el artículo [56] que comparan los resultados de evaluar ejercicios a través de diferentes algoritmos. En el estudio se deduce que el DTW, es el algoritmo que obtiene mejor precisión, incluso en presencia de ruido siempre y cuando las series a evaluar sean cuantitativas.

Dicho esto ahora solo faltaría encontrar una relación que vincule los ejercicios que se realizan desde el hogar, y los datos de entrada del algoritmo. Esto se puede realizar de múltiples formas entre las que se podrían destacar el uso de cámaras en el entorno del paciente para recopilar información sobre la posición de sus articulaciones, o sensores de movimientos integrados en la vestimenta del paciente.

3.3. REHABILITACIÓN FÍSICA DE PACIENTES

3.3.1. Estudio del conocimiento

La historia de la especialidad en rehabilitación física tiene su origen en Estados Unidos durante el comienzo del siglo XX [34]. Fue desarrollada por el Dr. Frank Krusen el cual sufrió tuberculosis desde joven, y detectó que las personas que no realizaban actividades físicas tenían más probabilidades de recaer en la enfermedad. Tras años de darse cuenta de la relación entre la actividad física y la rehabilitación, fundó el primer departamento Académico de Medicina Física en la Escuela de Medicina de Temple.

La especialidad de la rehabilitación física llegó a Europa en 1958 con la fundación de la organización no gubernamental Unión Europea de Especialidades Médicas (UEMS). No fue hasta 1968, cuando en Ginebra un Comité experto de la Organización Mundial de la Salud reconoció la rehabilitación física como una especialidad a nivel mundial.

La rehabilitación física ha sido un gran avance a la hora de tratar un amplio abanico de enfermedades. El problema de la rehabilitación física es que las estadísticas demuestran que muchos de los pacientes no llegan a llevar a cabo correctamente la terapia de rehabilitación [16]. Hay diferentes tipos de rehabilitación en función de la duración de la terapia que es asignada al paciente.

- **Enfermedades agudas:** Al ser enfermedades con un corto periodo de duración, solo el 20 % de los pacientes terminan desistiendo de los ejercicios de rehabilitación.

- **Enfermedades crónicas:** En el momento en el que aumenta el tiempo de la enfermedad, aumenta proporcionalmente la tasa de abandono de la realización de los ejercicios. En caso de enfermedades crónicas, el 45 % de los pacientes dejan de realizar los ejercicios asignados por el terapeuta.
- **Cambios en los hábitos o en los estilos de vida:** En prácticamente el total de los casos en los que se estipulan ejercicios de rehabilitación rutinarios, el paciente termina por dejar de realizar los ejercicios asignados.

Tras conocer estos resultados surge la necesidad de incentivar al paciente a seguir con la terapia y reducir está tasa de abandono. En la siguiente sección se expondrán una serie de soluciones tecnológicas que persiguen este fin, ya sea a través de juegos, o mejorando la experiencia del paciente durante la rehabilitación.

3.3.2. Soluciones tecnológicas similares

Actualmente hay soluciones tecnológicas diseñadas en otras instituciones que proporcionan una función similar a la propuesta en este proyecto. A continuación se listarán las más relevantes.

Figura 3.6: Soluciones para ejercicios de rehabilitación remota
http://www.esclerosismultipleeuskadi.org/rehabilitacion-virtual/

- **TOyRa:** Haciendo referencia al artículo [17], Toyra es un sistema basado en computación en la nube capaz de captar un movimiento en tiempo real mientras utiliza realidad virtual para la rehabilitación motriz de pacientes desde casa. Está siendo utilizado en el programa de telerrehabilitación del Hospital de Parapléjicos de Toledo. A diferencia del proyecto propuesto, este sistema está basado en el sistema Kinect de Microsoft junto con una serie de sensores incorporado en el dispositivo. Lleva siendo utilizada desde 2011 y los resultados que está obteniendo son muy buenos.
- **BioTrack:** Como es mencionado en el artículo [33], es un sistema de realidad virtual para la rehabilitación de pacientes con daños cerebrales. Los desarrolladores realizaron un estudio que incluía a diez pacientes con hemiparesia crónica y participaron durante 20 sesiones con

el sistema BioTrack. El estudio reveló una significativa mejora entre la valoración inicial y la final. Esto revela que los sistemas de rehabilitación virtual son efectivos a la hora de realizar tratamientos a pacientes. El dispositivo está compuesto por una pantalla panorámica, un sistema muy similar al dispositivo Kinect y tres cámaras que detectan la posición del usuario a través de unas marcas reflectantes colocadas en determinadas partes del cuerpo.

- **Teleictus:** En el artículo [25] demuestran que existen desigualdades respecto a la atención sanitaria recibida por los pacientes. Está desigualdad es debida a problemas de cobertura que sufren los pacientes que viven en lugares de acceso más dificultoso. Por esto la telemedicina aplicada al ictus es planteada como una herramienta que resolvería este problema, permitiendo a todas las personas obtener los mismos avances durante su rehabilitación. Teleictus ha permitido mejorar notablemente el acceso a estas personas logrando resultados similares a los de un tratamiento convencional. Este proyecto ha sido desarrollado en el Servei de Salut de les Illes Balears.

- **Sistema inalámbrico de telemedicina:** Haciendo referencia al artículo [42]. Se describe un sistema para implantar un sistema de monitorización mediante el uso de Bluetooth y GSM que permita realizar un seguimiento a los pacientes. Este sistema es capaz de enviar electrocardiogramas y valores sobre la tensión arterial. Para ello es necesario implantar en el hogar del paciente una PAN (Personal Area Network) formada por el sistema encargado de monitorizar.

- **Sistema de reconocimiento de acciones mediante cámaras con detección de profundidad:** Este proyecto ha sido desarrollado como trabajo de fin de máster [55]. Este sistema se utiliza en el campo de la monitorización, reconocimiento de movimientos y tiene múltiples aplicaciones. Los componentes hardware que han sido utilizados son una cámara para detectar la profundidad y un kinect de Microsoft. Posee una red neuronal en la que se delega la tarea de reconocer los movimientos. Utiliza una red neuronal Fuzzy ARTMAP y en base a una serie de métricas es capaz de reconocer patrones.

3.4. DESARROLLO DE APLICACIONES EN DISPOSITIVOS ANDROID

3.4.1. Introducción

Haciendo referencia al libro de Robledo Fernández, D. (2014). Desarrollo de aplicaciones para Android II. Ministerio de Educación [49]. Android es un sistema operativo que fue diseñado por Google. En un primer momento para dispositivos móviles. Actualmente se puede encontrar en una gran cantidad de diversos dispositivos que desempeñas tareas completamente diferentes. El uso se Android en diferentes dispositivos ha aumentado debido a la llegada del Internet de las Cosas.

Android está basado en Linux y por tanto el núcleo del sistema operativo es software libre. Las aplicaciones se desarrollan usando una variación de Java denominada Dalvik y proporciona las interfaces necesarias para acceder al hardware del dispositivo. La figura 3.7 muestra el diagrama que representa como se organiza la arquitectura de Android entre las distintas capas que componen el sistema operativo.

Debido a que Android es una plataforma de código abierto, los desarrolladores han podido crear fácilmente aplicaciones de una manera rápida. El problema de esto es que instalar aplicaciones hechas por parte de terceros crea serios problemas de seguridad. Es por esto que generalmente las aplicaciones se comercializan de manera on-line. Hay varios mercados virtuales que permiten adquirir aplicaciones. El más grande y popular en dispositivos con este sistema operativo es Google Play.

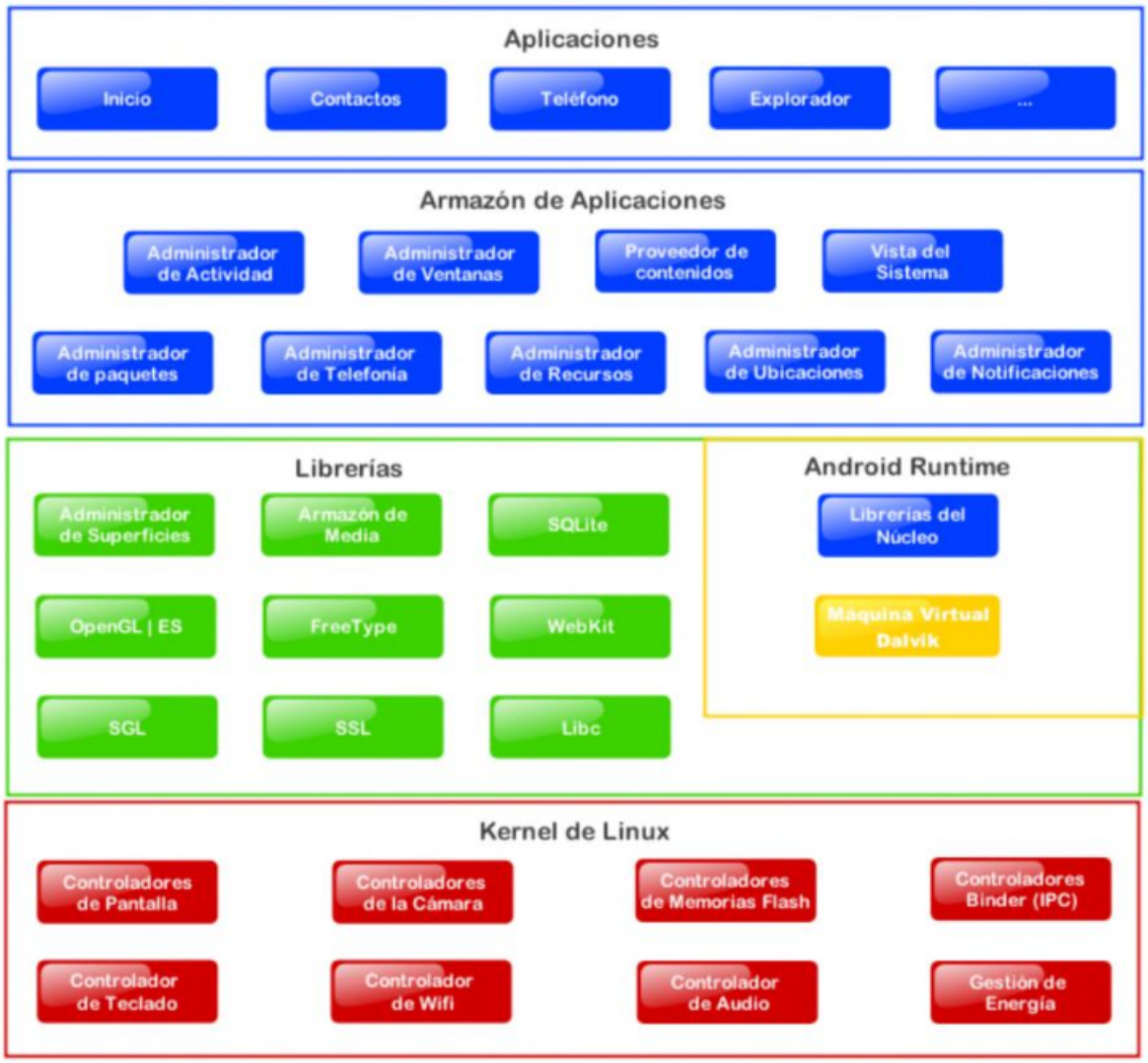

Figura 3.7: Diagrama de la arquitectura de Android
http://sedici.unlp.edu.ar/handle/10915/45851

3.4.2. Características de una aplicación Android

Como es indicado en el artículo Plataformas para el desarrollo de aplicaciones móviles [4] Google ha definido una estructura con los componentes más relevantes de una aplicación Android:

- **Activity:** Es el componente encargado de mostrar la interfaz gráfica al usuario. Una actividad es equivalente a una ventana en un entorno de aplicaciones de escritorio. Se debe definir una interfaz por cada actividad del proyecto. Los elementos que contiene una actividad deben quedar definidos en el fichero .xml que lleva asociado una actividad.
- **Listeners:** Son mensajes que producen un cambio de estado al ser recibidos por actividades o servicios. Está es la forma de unir componentes diferentes dentro de la misma aplicación. Los listeners son utilizados cuando ocurre un evento, y reaccionan mediante una determinada acción definida por el desarrollador.
- **Views:** Son los componentes de la interfaz de usuario, diferentes vistas pueden agruparse a través de grupos logrando una jerarquía, esto se logra a través de la disposición de los componentes a través de un archivo XML.

- **Service:** Ejecutan operaciones en segundo plano y carecen de interfaz. Un ejemplo de este componente podría ser la recepción de un mensaje por Whatsapp cuando el móvil está bloqueado.
- **Content Provider:** En este componente se delega la acción de almacenar información en el sistema operativo que va a ser compartida para que otras aplicaciones la usen. A modo de ejemplo, en este mismo proyecto, como se verá más adelante, se utiliza. Nuitrack tiene un content provider que proporciona información de gran valor a la aplicación que utilizará el paciente. Este componente tiene métodos estandarizados que permite a otras aplicaciones consultar, guardar o modificar información general.
- **Manifest:** Es un archivo fundamental en cualquier aplicación Android. En él se almacena la configuración de la aplicación, actividades que forman parte de la misma y permisos.
- **Broadcast Receivers:** Son componentes encargados de responder a advertencias y mensajes de difusión. Un ejemplo de este tipo de avisos podría ser batería baja o una llamada entrante.

3.4.3. Desarrollo de aplicaciones Android con Unity

El desarrollo de aplicaciones en Android precisa de un enfoque muy similar al del desarrollo en iOS. Sin embargo, por limitaciones propias del hardware, existen algunas diferencias características entre la versión de Android en Unity respecto a la de iOS.

Para poder desarrollar aplicaciones Android, es necesario tener un entorno de desarrollo previamente configurado. Esto implica instalar el SDK de Android con las diferentes versiones de Android con las que se pretenda trabajar. También es necesario o bien preparar un dispositivo físico, o una maquina virtual Android en el equipo.

Es habitual trabajar con el lenguaje de programación Csharp en Unity. Sin embargo, Unity Android permite invocar funciones implementadas en C/Cpp directamente desde los scripts en Csharp. Unity permite el desarrollo de aplicaciones más allá de los videojuegos. Como es indicado en el libro Learning Unity Android Game Development [13], la interactividad está siendo la clave del éxito empresarial. En los últimos años ha crecido el catálogo de productos que hacen uso de la Realidad Aumentada en entornos de desarrollo 3D. Estos productos pueden estar destinados a diferentes usos, ya sea para usos recreativos o con fines profesionales. Este es el momento en el que Unity se convierte en un poderoso aliado en esta labor.

Las posibilidades que ofrece Unity a la hora de crear entornos interactivos son enormes. Además Unity cuenta con un poderoso editor que permite crear interfaces atractivas para los usuarios. Esto permite mejorar la experiencia del usuario, sobre todo en personas que no están asociadas a novedades tecnológicas. Uno de los principales inconvenientes de esta plataforma es que no permite empezar desde una plantilla, sino que obliga a empezar desde cero e ir dando matices en cada detalle. Esto es debido a que Unity es un motor gráfio de proposito general, es por esto que no da nada hecho. Por otro lado, desde un punto de vista gráfico no es el que mejores prestaciones proporciona, ya que está a la cola respecto a motores gráficos como UDK. Aunque por otro lado admite un desarrollo muy sencillo en proyectos para smartphone. El hecho de ser demasiado complejo podría convertirse en un inconveniente.

Finalmente cabe a destacar que Unity permite publicar en diferentes plataformas como Play Store de Android en la que los usuarios podrían descargar las aplicaciones desarrolladas con este motor.

CAPÍTULO 4

METODOLOGÍA

En este capítulo se expondrá la forma en la que se ha desarrollado este proyecto dentro de unos medios y un contexto. Se explicará la metodología, el entorno y los componentes que se ha usado. Para ello se ha definido una metodología, ya que el uso de metodologías durante el desarrollo de cualquier software ofrece una serie de ventajas:

- Facilita la planificación del trabajo.
- Mejora el control y el seguimiento sobre el desarrollo.
- Mejora la gestión de los recursos disponibles durante el proyecto.
- Agiliza la comunicación con el cliente.
- Permite definir el tiempo y la calidad del desarrollo.
- Estipula un ciclo de vida adecuado para el proyecto.

4.1. ELECCIÓN DE LA METODOLOGÍA A SEGUIR

Se ha decidido utilizar una metodología ágil, ya que haciendo referencia al libro "Metodologías ágiles en el desarrollo de software."[5], las metodologías ágiles son una alternativa a metodologías tradicionales más novedosa, que se adapta a las actuales características del mercado.

La necesidad de aplicar una metodología ágil deriva del tiempo disponible a la hora de desarrollar el proyecto. Además de que es una buena práctica en proyectos que están obligados a realizar muchas modificaciones en poco tiempo, sin que está perdida de tiempo obligue a reducir la calidad del resultado. El uso de metodologías ágiles ofrece una serie de ventajas:

Dentro de las metodologías ágiles hay una amplía gama de metodologías, que está en constante evolución en los últimos años, y que cumplen con todos los requisitos para ser metodologías ágiles. Para este proyecto se ha utilizado **eXtreme Programming** ya que aporta una serie de características que hace de ella una metodología idónea para el alcance de este proyecto.

Al ser un proyecto a largo plazo, eXtreme Programming es muy eficiente durante el proceso de pruebas y planificación, ademas su tasa de error es muy baja. Esta metodología deriva en una programación muy organizada y fomenta la comunicación entre el desarrollador y el cliente. Una de sus ventajas es que se puede aplicar a cualquier lenguaje de programación, y ya que este proyecto combina el uso de varios lenguajes, hace que está metodología sea la más adecuada.

4.2. EXTREME PROGRAMMING

Como es indicado en el artículo de Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP) [32], la clave de esta metodología ágil es la colaboración entre el equipo de trabajo. Está orientada a incrementar el trabajo en grupo contribuyendo a que los desarrolladores aprendan más rápido unos de otros. Existe una comunicación bidireccional y fluida entre el cliente y el equipo de trabajo. Normalmente es utilizada en proyectos de una naturaleza inestable en la que se realizan constantemente cambios. También promueve el desarrollo más eficiente ahorrando tiempo sin perder calidad.

La Extreme programming se basa en varios componentes principales que se utilizan constantemente para dirigir el desarrollo del software y agilizar la comunicación con el cliente.

Por un lado, está metodología utiliza una especie de tarjetas para la comunicación entre el cliente y el equipo de desarrollo (en este caso es el desarrollador de este proyecto). Estás tarjetas son denominadas historias de usuario, son utilizadas para especificar los requisitos del cliente. Estás tarjetas son definidas antes de comenzar el desarrollo, aun que tras su inicio, es posible que durante algún punto sean modificadas por razones dispersas. Están compuestas por varios campos entre los que debe incluir como mínimo el nombre de la tarea a realizar y una descripción sobre que consiste la tarea.

Lo más común es que una historia de usuario dure entre una y dos semanas. Habrá casos en los que los clientes no ajusten de una manera adecuada el tiempo de desarrollo que podría llevar una determinada tarea. En este caso, se debería fragmentar la tarea en dos subtareas más simples que permitan realizarlas en el tiempo estipulado.

Durante el desarrollo de esta metodología se avanzará en el proyecto a través de la realización de iteraciones[32]. Antes del comienzo de cada iteración el cliente ha de reunirse con el desarrollador para aclarar posibles cambios que haya que realizar en las historias de usuario. Tras realizar o no estás modificaciones, el desarrollador iniciará la iteración y descompondrá las historias de usuario en tareas que irá realizando.

Es habitual que se definan diferentes funciones durante esta metodología ágil, pero en este caso particular solo hay dos participantes, el desarrollador del proyecto y el tutor jugando el rol de cliente.

El desarrollador es el encargado de implementar las historias de usuario. Una vez que obtiene el código, al tratarse de una única persona, es el encargado de realizar las correspondientes pruebas. En un primer momento debe comprobar que todos los elementos funcionan bien a nivel individual. Posteriormente, debe integrar los nuevos componentes en el sistema multi-agente y comprobar que la integración en el sistema se ha realizado adecuadamente.

Por otro lado, el cliente es el encargado de revisar cada historia de usuario al principio de cada iteración. En el caso particular de este proyecto, es el propio cliente el experto en el uso de esta metodología. Esto es una ventaja ya que es más improbable que una persona experta en este campo deje mal balanceada la carga de trabajo de cada historia de usuario. Si el cliente decide rectificar la trayectoria del proyecto introduciendo algún cambio nuevo debe seguir una serie de pasos.

Primero debe contactar con el desarrollador para especificar los cambios que va a realizar en las historias de usuario antes de que se produzca la siguiente iteración. Después hay que volver a ajustar la carga de trabajo, ya que en función de las nuevas peticiones del cliente, puede ser que haya que añadir varias historias adicionales. Al igual que se puede dar este caso, también podrían reducirse el número de historias de usuario en el caso de que el cliente prescinda de alguna funcionalidad del proyecto.

4.2.1. Fases de Extreme Programming

El ciclo de desarrollo de este proyecto ha supuesto una serie de pasos que serán explicados a continuación [26].

En una primera fase tras una reunión, el cliente definirá las historias de usuario que serán necesarias para realizar la primera entrega. Mientras el desarrollador investigará como va a ser la arquitectura del sistema, cuales podrían ser las tecnologías que mejor se adapten en el contexto del proyecto y una serie de pruebas para determinar cual escoger finalmente.

Durante la reunión el cliente determinó que historias de usuario son más relevantes para la primera entrega. El desarrollador hizo una estimación del tiempo necesario para realizar la primera entrega. Al estar un único programador en el proyecto, las historias se han ido programando constantemente de forma secuencial. En esta fase también se realizó una estimación sobre el tiempo que debe llevar el proyecto para que sea rentable.

Tras la realización de la primera entrega se procede a realizar entregas compuestas por iteraciones de corta duración. En esta primera entrega ha quedado definida la estructura que tendrá el proyecto como especificó el cliente. En las siguientes iteraciones se empieza a integrar la funcionalidad al sistema bajo la prioridad por historias de usuario indicado por el cliente.

Tras la finalización de las iteraciones de desarrollo, se inicia la fase de pruebas en la que se verifica que el sistema cumpla con los requisitos de calidad. Una vez verificado que los cumple, se procede a poner el sistema en un entorno real. Este proyecto aún no ha iniciado la fase en la que es sistema pasa a producción. Aun así, es necesario indicar que tras ponerlo en producción requiere un mantenimiento mensual por posibles actualizaciones que se quieran incorporar, hasta el fin del ciclo de vida del proyecto.

4.3. COMPONENTES SOFTWARE Y HARDWARE UTILIZADOS DURANTE EL DESARROLLO

Antes de describir los componentes que han sido necesarios durante el desarrollo de este proyecto, se va a realizar una breve introducción sobre el entorno del proyecto.

El entorno de ejecución de este proyecto consiste en una cámara TVIco o cualquier dispositivo Android que soporte el software de Nuitrack, un sistema software multi-agente, y un ordenador personal. Lo único que necesita el paciente es un dispositivo Android que soporte este software, para tener una referencia, móviles como el Samsung Galaxy S8, ya soportan el software de Nuitrack y por tanto, es probable que en cuestión de pocos años, cualquier dispositivo sea capaz de utilizarlo. Esto implica que para utilizar este sistema el paciente solo debe hacer uso de su dispositivo móvil o Android dotado de una cámara.

Esta cámara es capaz de detectar los movimientos realizados por el usuario y un sistema de reconocimiento por voz que permite el control de parte de la solicitud mediante el uso de comandos por voz.

4.3.1. Componentes Software

A continuación se muestra en la tabla ?? un listado con todos los componentes software necesarios para el desarrollo de este proyecto.

Software	**Utilidad**
Ubuntu 18.04	Usado para el desarrollo del servidor
Windows 10	Usado para el desarrollo de los clientes
Python	Lenguaje de programación del servidor y la aplicación web
Csharp	Lenguaje de programación de los clientes
R	Lenguaje de programación para procesar series temporales en el servidor
HTML	Para crear la estructura del cliente web
Bootstrap	Para dar estilos al cliente web
KnockOutJS	Marco para utilizar JavaScript en el cliente web
ChartJS	Para mostrar gráficos en el cliente web sobre la evolución de un paciente
ZeroC Ice	Para desplegar el entorno multi-agente
Flask	El marco utilizado por el servidor web
Rserve	Marco para hacer peticiones al lenguaje de programación R desde otros lenguajes
Unity	Para desarrollar las aplicaciones Android
Android Studio	Utilizado para compilar las aplicaciones Android desarrolladas en Unity
Aplicación Nuitrack versión Pro	Necesaria para la interacción con la aplicación del cliente, ya que este software realiza el tracking por la cámara.

Tabla 4.1: Software utilizado durante el desarrollo del proyecto

4.3.2. Componentes Hardware

A continuación se muestra en la tabla ?? un listado con todos los componentes hardware necesarios para el desarrollo de este proyecto.

Hardware	**Utilidad**
Portatil Asus RoG	Usado para el desarrollo del servidor, la aplicación web, y los clientes. En él se han desplegado el servidor, el servidor web y el servidor Rserve.
Orbec Persee	Dispositivo utilizado para la ejecución de la aplicación cliente que utilizará el usuario final.

Tabla 4.2: Hardware utilizado durante el desarrollo del proyecto

CAPÍTULO 5

ARQUITECTURA

Ahora se comentará el ***Sistema Multi-agente para Rehabilitación Física de Pacientes con Enfermedades Neurológicas*** desde un punto de vista más técnico. En primer lugar se describirá el sistema sin entrar en mucho detalle para tener una visión general de su estructura. Posteriormente se empezará a entrar en detalles.

5.1. VISIÓN GENERAL

El desarrollo de este trabajo de fin de grado ha supuesto la implantación de un sistema compuesto por diferentes agentes que trabajan sincronizados entre si y concurrentemente para ofrecer resultados, teniendo en cuenta las limitaciones propias de dispositivos Android y de un ordenador convencional, evitando así problemas de latencia y de rendimiento a la hora de realizar los cálculos.

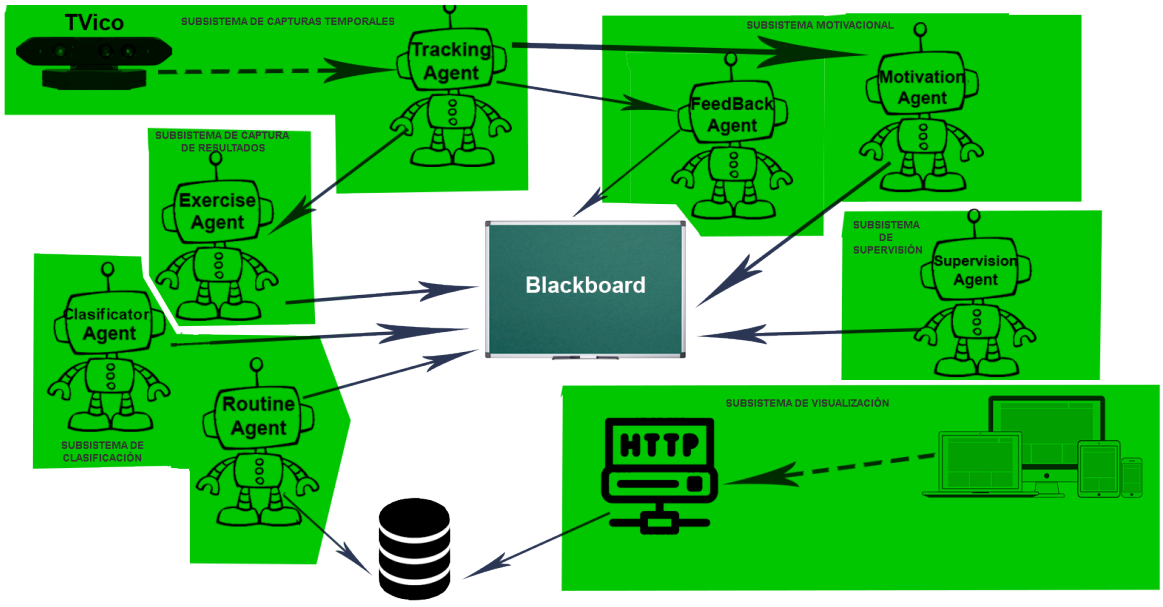

Figura 5.1: Sistema completo seccionado en clientes android, servidor descompuesto entre los diferentes agentes, base de datos y parte web

Los componentes más relevantes del sistema son los **clientes del sistema**, por un lado el que el paciente utilizaría y por otro el que utilizaría el terapeuta o administrador del sistema para actualizar movimientos nuevos en la base de datos, el **servidor** que realiza todos los cálculos y da lógica a la aplicación del cliente, y la **interfaz de administración web**, que sería usada por el terapeuta.

5.1.1. Clientes Android

Tanto el cliente administrador como el que utilizará el paciente son una parte fundamental del sistema. Hacen peticiones a los agentes que componen el servidor con el objetivo de obtener información de valor para el usuario que lo está utilizando. Existe un intercambio de datos constante entre el servidor, y estos clientes de manera bidireccional que permite múltiples acciones.

En este apartado se explicará como interpretan los clientes todas las articulaciones que son recuperadas a través de la cámara del paciente. El dispositivo que ejecuta ambos clientes ha de tener instalada una aplicación llamada Nuitrack. Está aplicación actúa como servidor con el objetivo de enviar datos a las aplicaciones clientes que representan el cuerpo del paciente.

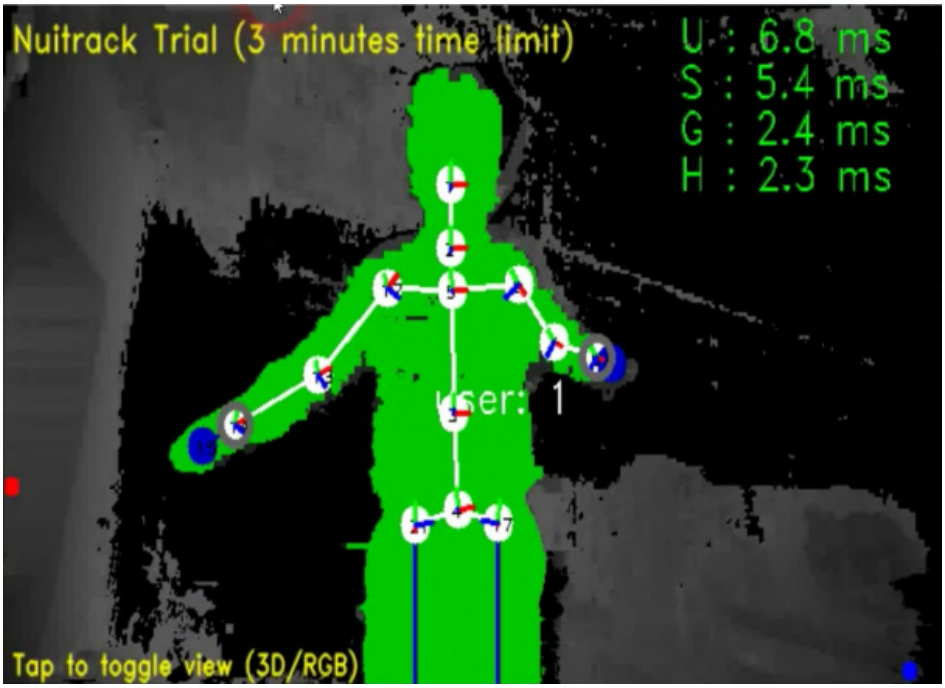

Figura 5.2: Nuitrack: La aplicación que envía datos del tracking a las aplicaciones clientes.

Nuitrack realiza un tracking al cuerpo del usuario que se ubique delante de la cámara reconociendo las coordenadas y la rotación de las articulaciones. También reconoce las conexiones entre las articulaciones dando como resultado una especie de esqueleto del paciente. Por otro lado, también puede reconocer gestos y expresiones faciales, aun que estás funciones aún están en una versión beta para la mayoría de dispositivos.

Para el correcto funcionamiento de este sistema solo ha sido necesario reconocer las coordenadas tridimensionales de cada articulación del sistema, y los gestos realizados con las manos. Todos estos datos los clientes los recuperan realizando llamadas a través de métodos a la aplicación Nuitrack. En la figura 5.3 se muestran las articulaciones que son recopiladas tras hacer las correspondientes llamadas a Nuitrack.

Después codifican los datos y se envían al servidor. Las coordenadas tridimensionales de cada articulación son utilizadas en el servidor para reconocer que movimiento se ha hecho. Después se filtran las articulaciones más relevantes al realizar un movimiento y se analizan para ver como de bien hecho está el movimiento. Esto se explicará en apartados posteriores.

5.1.2. Servidor

El servidor se podría dividir en los agentes que forman parte del sistema.

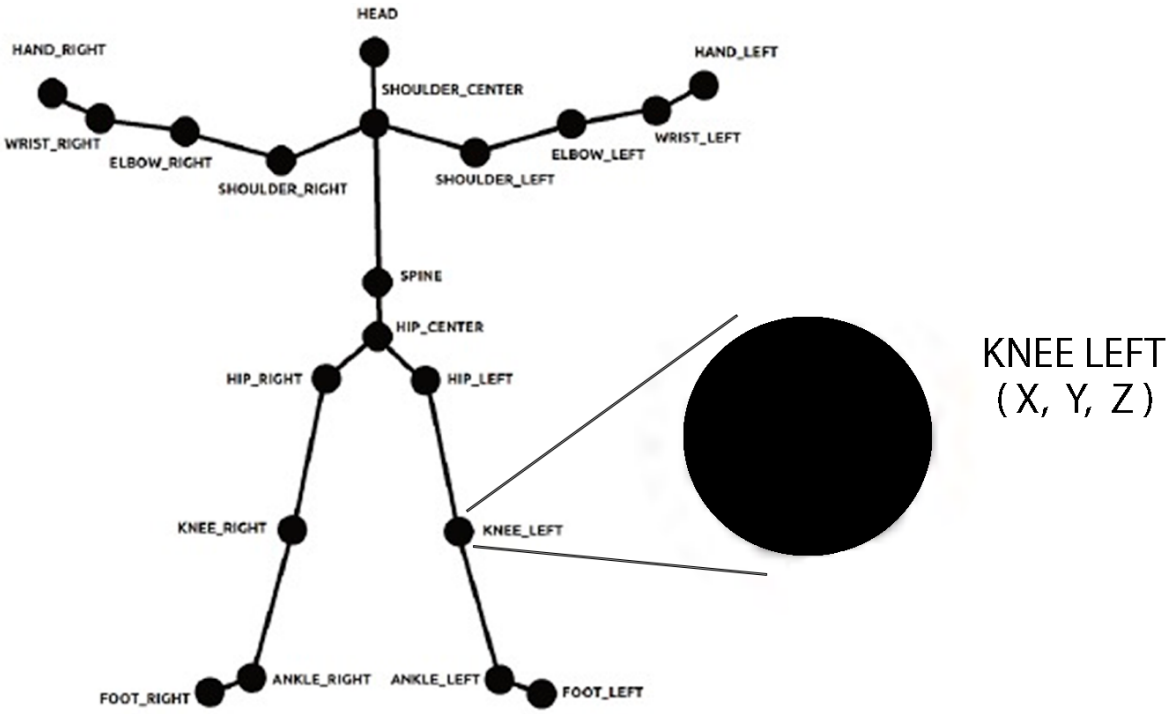

Figura 5.3: Nuitrack: Articulaciones que reciben los clientes de Nuitrack.

- Subsistema de capturas temporales. Este subsistema, compuesto por la cámara del paciente y por el tracking Agent del sistema, es el encargado de recopilar la información de los movimientos que realiza el paciente, o instructor para su posterior uso. En el caso de que el movimiento lo haga el paciente, el tracking Agent enviará las coordenadas del esqueleto capturadas por la cámara al subsistema de supervisión, para su posterior comparación a través de algoritmos que serán explicados posteriormente. En el caso de que el movimiento sea realizado por un instructor, este subsistema se comunicará con el subsistema clasificador para indicar que movimiento ha realizado y este se encarga de almacenarlo en una base de datos para su posterior uso en comparaciones con movimientos que podrían estar bien hechos o no.

- Subsistema de captura de resultados. Una vez finalizado el movimiento del paciente, este componente del sistema se encarga de comunicarse con el subsistema de supervisión para recopilar la evaluación del paciente, y poder mandar los resultados, tanto a la interfaz del paciente que está utilizando la cámara, como a la persona cualificada de monitorizar la evolución del paciente.

- Subsistema de visualización. Este subsistema tiene delegada la tarea de interactuar con los usuarios del sistema, indistintamente del rol que juegue en él.

- Subsistema de supervisión. Una vez identificado el movimiento que está realizando el usuario, el subsistema de supervisión utiliza las series temporales de las articulaciones que necesite en función del movimiento que ha realizado el paciente, ya que no tiene sentido utilizar las articulaciones de la pierna, cuando comparamos un movimiento del brazo, y las compara con series representativas, series que están realizadas correctamente por el médico que imparte la terapia, para evaluar como de bien ha realizado el movimiento el paciente y poder procesar los resultados posteriormente.

- Subsistema de clasificación. Es uno de los componentes del sistema que más recursos consume, ya que se encarga de evaluar cual es el movimiento que acaba de realizar el usuario, y para ello requiere varias comparaciones con los movimientos almacenados en el sistema previamente.

A diferencia del subsistema de supervisión, en el cual solo se comparan las articulaciones de acuerdo al movimiento que se ha identificado, al desconocer que movimiento es el que el paciente ha hecho en un primer momento, es necesario analizar todas las articulaciones que el sistema recopila, una vez que el subsistema de clasificación ha identificado el movimiento, el resto de articulaciones que no intervienen en el son descartadas para que el subsistema de supervisión evalúe como de bien ha sido realizado el movimiento.

- Subsistema motivacional. Es un componente del sistema que podría haber sido una de las partes más destacadas del sistema, pero de momento solo va a ser una parte más. El motivo del comentario anterior es que en un primer momento este subsistema iba a realizar un reconocimiento facial del usuario que este usando el sistema en un momento dado, permitiendo analizar su edad y su estado de ánimo, permitiendo ofrecer una experiencia más personalizada. El problema es que los desarrolladores de la API de Nuitrack, el software que permite hacer el tracking del esqueleto, aún no ha lanzado la actualización que nos permite esta operación. Por tanto, actualmente esta parte del subsistema ofrece frases que animen al usuario a seguir usándolo, y le propone otros movimientos que podría realizar utilizando el sistema.

5.1.3. Cliente Web

De algún modo es necesario que todos los datos que el sistema va recopilando, puedan ser visualizados. Estos datos son interpretados por el cliente web. Este cliente será utilizado por el terapeuta. Permite ver los últimos usuarios que han utilizado el sistema, y monitorizar los movimientos que están realizando remotamente. De esta forma podemos monitorizar la rehabilitación de cada paciente de manera individual.

Además en función de cada caso, es probable que algunos pacientes tengan que realizar varios movimientos diferentes. Con esta herramienta se pueden monitorizar solo determinados movimientos, o tener una visión global de la rehabilitación del paciente. Gracias a ello, es posible detectar si el paciente sufre más dificultades al realizar un determinado movimiento, y de este modo sugerirle alguno equivalente que sea menos tedioso de realizar.

5.2. DISEÑO DE LA SOLUCIÓN PROPUESTA

A continuación se planteará el problema que se ha resuelto, el modo para llegar a la solución del problema y las ventajas que aporta está solución.

Por un lado, ha sido necesario proporcionar a los pacientes una herramienta que pueda servir de apoyo a ellos y al terapeuta durante las sesiones de rehabilitación. Al ser una herramienta que utilizarían personas adultas de un amplio rango de edades, es necesario que está herramienta sea fácil de utilizar.

La solución a este problema ha sido incorporar el sistema que ha permitido realizar reconocimiento de esqueletos de Nuitrack. Es una solución más eficaz que otras soluciones equivalentes, como podría ser Kinect en términos de rendimiento. Tras realizar una comparación, se ve a simple vista que Nuitrack reconoce de una forma mucho más eficaz el esqueleto. Esto supone una ventaja ya que dota a la herramienta de una mayor precisión.

Además la herramienta que se ha utilizado finalmente reconoce gestos, y solo en algunos dispositivos expresiones faciales. Esto abre todo un campo de investigación y posibilidades de mejora de la herramienta. Permitiendo escalar mucho más un sistema que trabaje con las tecnologías utilizadas en este proyecto, que las soluciones equivalentes de competidores.

Tras poner solución a este problema, era necesaria una plataforma que se encargará de todo el cómputo relacionado con el reconocimiento de los movimientos que realiza el paciente. Como se ha

mencionado antes es un sistema que podría ser utilizado por personas poco familiarizadas con la tecnología, y una forma más eficaz de simplificar el uso de la herramienta a los pacientes, es que el propio sistema sea capaz de identificar el movimiento que está haciendo.

El problema es que esto requiere la realización de múltiples tareas en paralelo para que el sistema funcione en tiempo real. Por tanto el uso de un sistema multi-agente es una solución idónea para un proyecto de estas características. Para montar el entorno en el que se desplegará, se ha decidido utilizar ZeroC Ice por su alta flexibilidad a la hora de diseñar sistemas heterogéneos.

Finalmente también había que mostrar de algún modo toda la información que el sistema captura del paciente al terapeuta. Para que este pueda proceder a aplicar sus conocimientos para impartir la rehabilitación a los pacientes. Dado que la aplicación que utilizaría el paciente requiere conexión a internet, y ZeroC Ice es una plataforma completamente compatible a la hora de desplegar un sistema que trabaje online, se ha decidido que la interfaz con la que el terapeuta realice la rehabilitación sea web.

Esto aporta grandes ventajas teniendo en cuenta la tendencia del mercado a la computación en la nube. Que la interfaz del terapeuta sea web permitiría que el terapeuta pueda acceder al sistema desde cualquier lugar. Esto permitiría contemplar la posibilidad de subir en un futuro todo el sistema a la nube.

5.3. SERVIDOR

A continuación se enumeraran todos los agentes del sistema, analizaremos su código e iremos descomponiendo la solución propuesta. Cada uno de los agentes que componen el servidor está implementado en el lenguaje de programación python 3.6, y utilizan Zero C Ice como medio de comunicación.

Antes de mostrar el código de los agentes que componen el sistema es necesario ver la interfaz que utilizan para cargar los métodos de objetos que utilizan otros agentes y entender que acciones puede hacer cada agente. El siguiente código muestra la interfaz, en este caso denominada **interfaz.ice 5.1**

Debido a que finalmente toda la parte web del sistema no se ha podido incluir en el subsistema de Zero C Ice, se hablará de ello en otro apartado. El servidor procesa los datos emitidos por los clientes que componen el sistema. Este se encarga de interpretarlos, almacenarlos y analizarlos. Los datos que le lleguen determinarán la salida del sistema. A continuación se comentará la función de cada uno de los agentes que forman el sistema servidor.

Listado 5.1: Métodos que pueden invocar agentes de otros agentes

```
// -*- mode:c++ -*-

// @author Alejandro Medina Jimenez

module interfaz {

  dictionary<string, Object*> ObjectPrxDict;
  sequence<Object*> ObjectPrxVect;

  interface RoutineAgent {
    string obtenerSerie(string id, string nombre);
    string obtenerUltimoEjercicio(string nombre);
    void anyadirEjercicio(string datax, string datay, int n);
    void anyadirSerie(string art, string serie);
    void anyadirResultado(string nombre, string mov, string nota);
  };

  interface SupervisionAgent {
    string filtrarMovimientoParaEvaluar(string mov, string serie);
    double distanciaOE(string v1, string v2);
  };

  interface Container {
    string anyadirUsuariosRegistrados(string mov);
    void actualizarMovimientos(ObjectPrxDict dict);
    string obtenerMovimiento(string mov);
    int obtenerIndice(string name);
    string obtenerIP();
    string obtenerMovimientosRegistrados();
    string obtenerUsuariosRegistrados();
  };

  interface FeedBackAgent {
    string obtenerMensajeUsuario(string mov);
  };

  interface ClassificationAgent {
    string clasificarMovimiento(string serie);
  };

};
```

5.3.1. Comunicación entre los agentes

Es evidente que para implantar un sistema multiagente es necesario un medio de comunicación entre los diferentes agentes que componen el sistema. La comunicación es la base para las interacciones y la organización entre los agentes. Entre las diferentes formas de interaccionar propuestas en este sistema entre los diferentes agentes se pueden destacar la pizarra compartida y el paso de mensajes.

La pizarra es una memoria compartida que permite a los agentes compartir información de interés dentro del sistema. Cuando se emplea un sistema de pizarra no hay comunicación directa entre los agentes. En este caso los agentes manipulan la pizarra y después otros agentes acceden al contenido actualizado. Por otro lado, el paso de mensajes permite a los agentes intercambiar información que es relevante en una determinada acción en la que no es necesaria la intervención de todos los agentes.

Por tanto, antes de proceder a explicar la función de cada agente en el sistema, sería conveniente comprender como se realiza la comunicación entre ellos a través de la plataforma de ZeroC Ice. Zero C Ice permite realizar invocaciones síncronas y asíncronas a través de diferentes medios de

comunicación como TCP, UDP, SSL o Bluetooth.

Realiza conexiones bidireccionales que permiten al servidor reutilizar la conexión establecida por un cliente con el objetivo de devolver la llamada. Ofrece funciones de seguridad potentes y fáciles de usar. También utiliza un protocolo binario compacto y eficiente para minimizar el consumo de CPU y ancho de banda, por lo que es una solución ideal para este sistema.

Para el uso de este framework es necesario definir una interfaz5.1 en la que queden reflejados los diferentes mensajes que pueden utilizar para realizar la comunicación los agentes. El problema de utilizar ZeroC Ice sin ninguna extensión es un proceso tedioso ya que el proceso de configuración es complejo.

Para solucionar esto, se ha instalado la interfaz gráfica del paquete que permite visualizar los nodos que hay en el sistema de una forma más sencilla. En la figura 5.4 se puede observar está interfaz. Además al utilizar la interfaz gráfica, el proceso de configuración se hace más sencillo para el desarrollador. Esto permite al desarrollador centrarse más en el desarrollo del proyecto y menos en la comunicación.

Los elementos más importantes para desplegar este sistema serían:

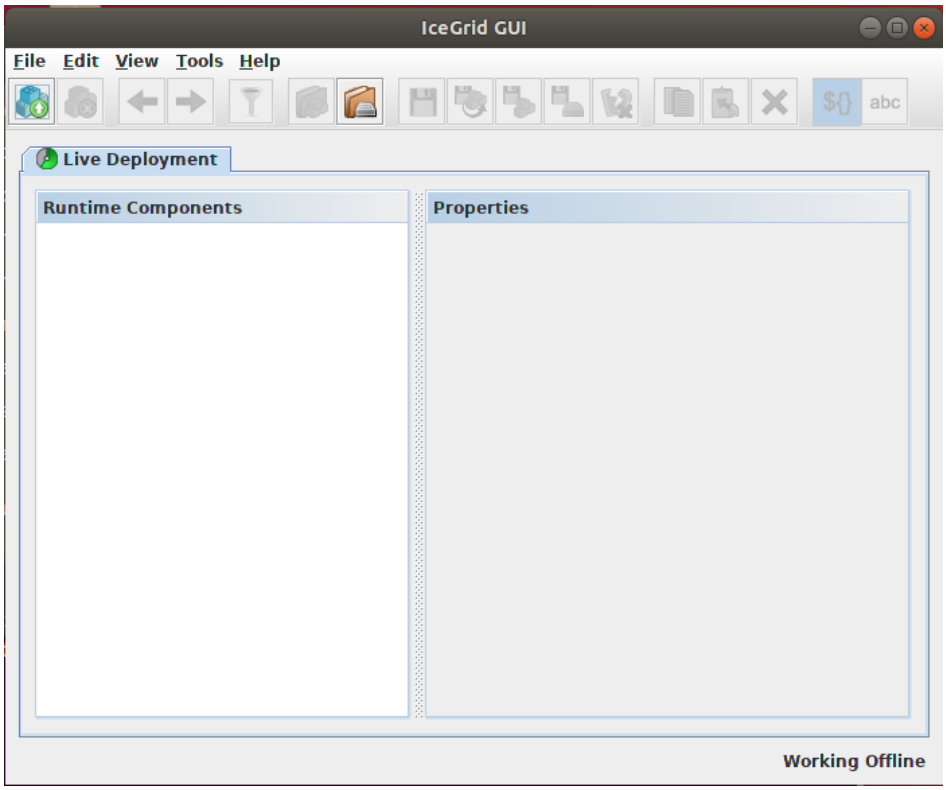

Figura 5.4: Interfaz gráfica en ZeroC Ice

- **Archivo de configuración (.xml):** Este fichero es generado automáticamente por la interfaz gráfica de ZeroC Ice. Los elementos más destacados son el nombre de la aplicación, que a su vez engloba todos los nodos del sistema. Cada nodo tiene estipulado un nombre y puede contener servidores a su vez. Los servidores tienen un identificador y una serie de parámetros, por ejemplo si la activación se realiza de forma manual, el nombre del agente que va a ejecutar y la ruta del mismo. Por otro lado, cada servidor tiene definidas una serie de propiedades. Las propiedades más utilizadas en el sistema son la salida de consola, la salida de errores, y la forma de recuperar los proxies de otros agentes del sistema.
- **Configuración del localizador:** Este fichero es fundamental, ya que contiene la configuración de red que indica donde está siendo ejecutado el sistema.

- **Configuración de cada nodo:** Contiene la configuración de cada nodo, es necesario un fichero por cada nodo que se genere en el sistema. Estos ficheros son pasados como parámetro durante la puesta en marcha de cada nodo. Los datos más relevantes son el nombre, la ruta donde se almacenarán los datos y los endpoints.

5.3.2. TrackingAgent

Listado 5.2: Objetos de Tracking Agent

```
1
2  class Coordenada():
3  def __init__(self):
4  self.x = ""
5  self.y = ""
6  self.z = ""
7
8  def anyadirValor(self, vx, vy, vz):
9  self.x = '{}{},'.format(self.x, vx)
10 self.y = '{}{},'.format(self.y, vy)
11 self.z = '{}{},'.format(self.z, vz)
12
13 def finalizar(self):
14 self.x = self.x[0:len(self.x)-1]
15 self.y = self.y[0:len(self.y)-1]
16 self.z = self.z[0:len(self.z)-1]
17
18 def obtenerSerie(self):
19 return '{}-,,-{}-,,-{}'.format(self.x, self.y, self.z)
20
21 class Articulacion():
22 def __init__(self, nombre):
23 self.nombre = nombre
24 self.vector = Coordenada()
25
26 def anyadirInstanciaArticulacion(self, vx, vy, vz):
27 self.vector.anyadirValor(vx, vy, vz)
28
29 def cerrarSerie(self):
30 self.vector.finalizar()
31
32 def toString(self):
33 return self.vector.obtenerSerie()
```

Este agente juega un rol muy importante en el sistema, ya se ha mencionado esto en varias ocasiones y probablemente el motivo sea que cuando en una comunidad de agentes, la carga de trabajo está bien balanceada, el papel de todos los agentes se hace de cierto modo en el sistema. En este caso, este agente se encarga de comunicarse con los clientes disponibles para dispositivos Android, ya sea la aplicación que utilizaría el terapeuta o la aplicación que utilizaría el paciente.

Para explicar su funcionamiento más básico haré uso del algoritmo de la figura 5.1. Antes de explicarlo, mencionar que la comunicación entre las aplicaciones Android y el sistema Zero C Ice se realiza por medio de sockets TCP. En un primer momento se determinó que podría utilizarse otra tecnología pero los plazos de entrega del proyecto determinaron que se mantendría la conexión con esta tecnología.

Este agente tras conectarse con el usuario recibe datos que pueden estar en diferentes formatos, y es tarea del agente interpretarlos para determinar que acciones tomar. Lo primero que se hace es comprobar si el cliente que ha iniciado su conexión ha enviado datos, en caso de que el paquete venga vacío se aborta la conexión.

Algoritmo 5.1: Gestión de datos del agente de tracking

```
   Datos     : Datos recibidos por el cliente
   Resultado : Diferentes acciones del agente
 1 while siempre do
 2     se asigna a la variable fin el valor falso Se inicia la conexión con un cliente;
 3     while mientras fin sea distinto de true do
 4         recibir datos del cliente;
 5         if no hay datos del cliente then
 6             se asigna true a fin;
 7         else
 8             se decodifican los datos;
 9             if si datos es igual a solicitud de movimientos en el sistema then
10                 se envian los movimientos que el agente container tiene registrados;
11             else
12                 if si datos es igual a solicitud de usuarios en el sistema then
13                     se envian los usuarios que el agente container tiene registrados;
14                 else
15                     if llega el fin de la serie que realiza el usuario then
16                         se cierra la serie realizada por el usuario;
17                         envía a los agentes del sistema el movimiento que ha hecho el
                            usuario;
18                         se asigna true a fin;
19                     else
20                         se interpretan las instancias representadas por numeros de la
                            posicion de un usuario en un instante de tiempo y se almacenan;
21                     end
22                 end
23             end
24         end
25         obtener serie del vector;
26     end
27 end
```

En caso de que el paquete contenga datos se procede a decodificarlos como un string y se identifica si el paciente ha realizado una solicitud para obtener los movimientos disponibles, usuarios disponibles o si ha iniciado la realización de algún movimiento. Tanto si ha solicitado movimientos disponibles, como si se ha solicitado usuarios disponibles la acción del agente de traqueo es recuperar estos datos del container que representa la pizarra del sistema y enviarle la información obtenida al cliente.

En el caso de iniciar un movimiento el cliente debe enviar como mínimo el nombre del usuario que realiza el movimiento, el movimiento ha realizar (este parámetro es opcional en el caso del cliente que tendrá el paciente) y una instancia en el tiempo de una serie de coordenadas representadas en forma de números que se descomponen en los diferentes ejes de las articulaciones del cuerpo de la persona que se encuentra delante de la cámara realizando el movimiento. Esta instancia que representa una posición de un usuario durante el movimiento se desglosa en los diferentes ejes que la componen y se almacenan en una posición de un vector, para ello se utilizan las clases descritas en el listado 5.2.

Este proceso se repite en varias iteraciones hasta que el cliente decide que el movimiento ha finalizado, en este momento el agente de traqueo recibe un valor que representa la finalización del movimiento. Este valor varía en función de si el movimiento se ha realizado con fines de almacenamiento para ser comparado con otros movimientos, o si se ha realizado con el objetivo de obtener una calificación por el movimiento.

En caso de que se haya hecho para ser almacenado y posteriormente utilizado como referencia, se codifica la serie en un formato para su almacenamiento y se llama al Routine Agent para que inserte esta serie en la base de datos. Por otra parte, en caso de que se haya realizado con el fin de que sea comparado y analizado, lo primero que hace este agente es una llamada al agente clasificador pasándole como parámetro la serie recuperada del usuario y recuperando tras esta llamada el movimiento que el sistema piensa que el paciente ha realizado.

Después, una vez que se conoce el movimiento que el paciente ha realizado, se envía el movimiento que el sistema ha determinado y la serie completa al agente supervisor, que se encargará de filtrar por las articulaciones relevantes a la hora de proporcionar una calificación, y tras realizar los correspondientes cálculos, devolverá una calificación. Esta calificación se le envía al Routine Agent para que la almacene en la base de datos del sistema.

5.3.3. RoutineAgent

Este agente es el encargado de manipular la base de datos. El resto de agentes le realizan peticiones para que inserte, modifique o borre datos en base a sus intereses. Era necesario un agente que realice esta función debido a que había que almacenar datos en el sistema como podrían ser los nombres, información sobre los pacientes, o las series que el sistema toma como referencia para analizarlas y compararlas. La solución propuesta es un agente que gestiona la base de datos implementado en python3.6 y con sqlite3.

Es necesario importar la librería sqlite3 que tiene una sencilla instalación. Gracias a utilizar sqlite, no es necesario tener una instancia de un servidor y por tanto el sistema es independiente. Además este sistema de datos es altamente confiable y ligero, y por tanto lo hace ideal para un sistema integrado que requiere de rápidos accesos. A continuación describiremos los métodos más relevantes de este agente.

El método descrito a continuación es **obtenerSerie**. Este método se utiliza tras obtener la serie que se va a proceder a clasificar y después a analizar, para obtener los movimientos de referencia que van a ser comparados con los movimientos que el paciente realice. Un movimiento se descompone en las articulaciones por ejes que la cámara reconoce, como la cámara tiene profundidad detecta las coordenadas x, y, z.

Por tanto, un movimiento está compuesto por un identificador, el movimiento realizado, y el nombre de la articulación asociada al movimiento que se almacena en tres ejes. El resultado de esta consulta es una coordenada de la articulación de la última serie almacenada en el sistema con intención de ser comparada por otra proporcionada por el paciente. El resultado está expresado en forma de vector compuesto por números en coma flotante.

Por otro lado, el método **obtenerUltimoEjercicio** sirve para obtener el último movimiento que ha realizado un usuario determinado. Este método es usado por el Agente Motivacional para obtener los datos necesarios para recomendar otro ejercicio al paciente.

El método **anyadirSerie** es utilizado por el cliente que posee el terapeuta para poblar la base de datos con nuevos movimientos. Esto sirve para rectificar ejercicios ya insertados, o crear ejercicios nuevos.

El método **anyadirResultado** es utilizado por el Agente Tracking para escribir la nota una vez que se ha reconocido el movimiento que ha realizado el paciente.

Gracias a utilizar un sistema multi-agente, sería muy sencillo incrementar la funcionalidad de este agente incorporando métodos nuevos en el hipotético caso de aumentar el número de tablas en la base de datos, en el caso de que por ejemplo, en una nueva actualización se incorpore reconocimiento facial al sistema.

5.3.4. SupervisionAgent

El agente supervisor juega un rol fundamental en el sistema ya que es el encargado de analizar el movimiento del usuario tras determinar de que movimiento se trata. Para realizar está tarea es necesario tener disponibles las dos instancias en el tiempo a comparar de cada una de las articulaciones involucradas en el movimiento.

Antes de explicar de donde son recuperadas las series que se comparan, es necesario entender como se realizan las comparaciones. Como ya se ha explicado previamente, las articulaciones están compuestas por unas coordenadas tridimensionales y un nombre. La forma en la que se han decidido comparar dos movimientos, es comparando las coordenadas de las articulaciones con el mismo nombre. De esta comparación se obtiene una nota que se explicará como es manipulada más adelante.

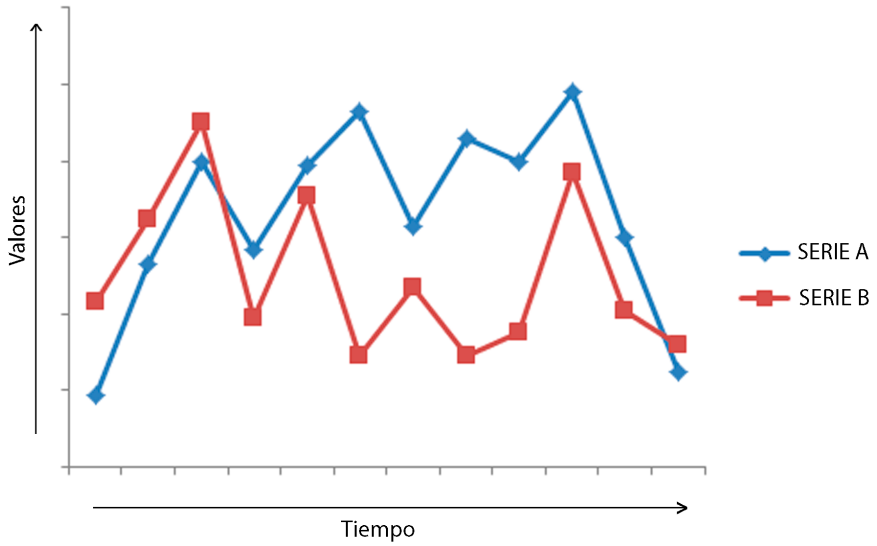

Figura 5.5: Ejemplo de comparación de una coordenada de una articulación

En la figura 5.5 se puede observar la comparación de dos series. En el caso de que se estuvieran comparando las articulaciones de la cabeza de dos movimientos distintos, habría que descomponer estos movimientos en tres coordenadas. Este podría ser un ejemplo de la comparación de la coordenada x de la articulación cabeza, de dos movimientos distintos.

Por otro lado, a través del Routine Agent podemos obtener los vectores de coordenadas de las articulaciones que representan un movimiento bien hecho en el contexto de un determinado movimiento. Por otra parte, debe ser el Tracking Agent el encargado de proporcionar estos datos recopilados y filtrados del movimiento del paciente. Para llevar acabo esta comparación se han ido resolviendo una serie de problemas que serán explicados a continuación.

La forma en la que se deciden comparar los movimientos, es comparando uno a uno los ejes de una todas las articulaciones involucradas en el movimiento. Por tanto en el hipotético caso de que se compare un movimiento en el que intervienen cuatro articulaciones del cuerpo, sería necesario realizar doce comparaciones entre las articulaciones representativas de un movimiento correcto, y las del paciente que realizó el movimiento.

En este punto surge un problema, ya que un movimiento no dura una cantidad de tiempo determinada, ya que puedes realizar un movimiento bien en tres segundos en una ocasión, y en otra ocasión distinta puede durar cuatro segundos y seguir estando bien hecho el movimiento. Por tanto surge la necesidad de comparar dos vectores interpretando que cada instancia de una coordenada capturada en el tiempo de una articulación sea una posición del vector. La solución propuesta a este problema es el uso del algoritmo de Deformación dinámica del tiempo (Dynamic Time Warping)[56]. Este algoritmo como ya se ha explicado antes permite obtener una correspondencia óptima entre dos secuencias a través de una distorsión no lineal con respecto al tiempo.

En este punto surge otro problema diferente, y es que como se ha indicado antes, el lenguaje de programación en el que ha implementado el servidor es python 3.6, pero tras una tediosa búsqueda se llegó a la conclusión de que no hay ninguna librería en python 3.6 que se adapte a las necesidades de este proyecto. Durante esta búsqueda se localizó otra libraría que si se adaptaba a las necesidades de este proyecto, pero estaba hecha para el lenguaje de programación R.

La solución a este problema fue posible gracias a la librería Rserve, que permite poner a la escucha un servidor R y hacerle peticiones con cualquier lenguaje de programación. Esto último vino bastante bien debido a que durante la fase de comparación y análisis del movimiento, es necesario realizar una gran cantidad de llamadas a la librería que contiene el algoritmo Dynamic Time Warping. Por tanto tener un servidor que puede ejecutar este algoritmo, permite balancear mejor la carga computacional de los diferentes agentes que forman el sistema.

A continuación se describirán las operaciones más relevantes que este agente puede hacer representadas por métodos.

El método **filtrarMovimientoParaEvaluar** muestra como este agente se centra exclusivamente en las secciones de la serie más relevantes en base al movimiento que se va a supervisar. Al estar el movimiento ya clasificado hay partes que en función del movimiento que se analice son irrelevantes.

Es muy probable que si vas a analizar un movimiento asociado a la cabeza la posición de las piernas tenga poca relevancia. Por tanto, desglosamos un movimiento en los diferentes vectores de los ejes de las articulaciones más relevantes que lo componen, tanto por parte del movimiento de referencia, como por parte del movimiento realizado por el paciente.

Después recorremos estos diferentes vectores con un bucle y los comparamos con el método distanciaDTW que será explicado a continuación y en base a las distancias de los diferentes vectores calculamos una nota que es retornada al agente que invocó este método en el sistema. El cálculo de la nota se realiza en el método calcularNota que básicamente realiza una media aritmética de las distancias.

El método distanciaDTW descrito en la figura 5.3 recibe como parámetros dos vectores rellenos

Algoritmo 5.2: Cómo calcular la nota de un movimiento

Datos : nombre del movimiento analizado y todos los vectores de articulaciones por ejes captados
Resultado : nota en base al movimiento que el sistema estima

```
Se transforma el movimiento en un vector con las articulaciones que lo componen;
while No es el fin del vector do
    leer movimiento en base de datos;
    obtener serie del vector;
    comparar distancias;
    añadir nueva distancia calculada al vector;
end
calcular nota;
devolver nota;
```

Algoritmo 5.3: Llamada Dynamic Time Warping

Datos : dos coordenadas de dos articulaciones de dos movimientos diferentes
Resultado : distancia entre dos series

```
nota = llamada Rserve para que haga los calculos pasando como parametro las dos series;
devolver nota;
```

con números en coma flotante de diferente tamaño. La variable callfunction almacena un String que representa la llamada a una función cargada previamente en la declaración de la conexión con Rserve.

Después se pasa como parámetro al método eval de conn2, que básicamente representa una instancia del servidor que responde a cálculos en R. Finalmente se retorna el valor resultante de la comparación de estos dos valores, siendo 1 el valor más óptimo que es posible obtener. A diferencia de lo que se pueda pensar en un primer momento, el valor más pequeño es el más óptimo, por tanto otro posible resultado de esta llamada representando una comparación pésima podría ser quinientos.

Listado 5.3: Método distanciaOE de Supervision Agent

```
1
2 def distanciaOE(self, serie1, serie2, current=None):
3   callfunction = "dtw_OE(c({}),␣↩
        ↪ c({}))".format(serie1[0:len(serie1)-1], serie2)
4   distancia = self.conn.eval(callfunction)
5   self.llamadas_dtw = self.llamadas_dtw + 1
6
7   return distancia
```

Finalmente será descrita la figura 5.3. El motivo por el cual se ha dejado para el final este método es que no aporta valor al sistema en el punto en el cual está siendo descrito. En un momento de la implementación, se delegó la tarea de comunicarse con el servidor R a este agente. Este método es utilizado por el agente clasificador, ya que este agente tiene un solo método pero una elevada carga de cómputo y esto nos permite balancear la carga entre los dos agentes.

5.3.5. Container

El container juega un rol de pizarra en el contexto de un sistema multiagente. En el que se almacena información de valor del sistema que los agentes pueden consultar, modificar, borrar y añadir. Los agentes lo usan a modo de repositorio de conocimiento para intercambiar la percepción que tienen del entorno. Tiene funciones adicionales como traducir un movimiento en las diferentes articulaciones

que intervienen en el movimiento. Incorpora usuarios registrados en el sistema y los diferentes movimientos disponibles en el sistema y que un paciente podría hacer. En el que se almacenan parámetros de configuración del sistema.

5.3.6. FeedbackAgent

Este agente es el encargado de interpretar la nota que ha obtenido el paciente tras realizar el ejercicio entre otras tareas que serán explicadas a continuación. Este agente tuvo que modificar su funcionalidad durante una de las iteraciones del desarrollo de este proyecto debido a factores externos que no estaban contemplados durante el desarrollo.

La idea principal para este agente era que fuese capaz de realizar un reconocimiento facial durante el reconocimiento de un movimiento del paciente que este haciendo uso del sistema. Esto sería muy útil para conocer el estado de ánimo del paciente y comprender de algún modo si el sistema puede mejorar, o incluso si le está gustando el tratamiento asignado. El problema es que esta clase de reconocimiento aún no está disponible para el dispositivo que se ha utilizado para realizar este proyecto, por tanto es una mejora sustancial que habría que posponer.

En los foros oficiales de Nuitrack la comunidad comentaba que es probable que en 2020 se resuelva este problema.

Algoritmo 5.4: Cómo calcular el mensaje de un paciente

Datos : Nota del movimiento realizado por el paciente
Resultado : Mensaje interpretando la nota del paciente

```
1  Se declara mensaje;
2  Se cambia el tipo de la variable evaluation a float;
3  if nota menor que 28 then
4  |   mensaje de motivacion para una nota muy buena;
5  else
6  |   if nota mayor que 28 y menor que 35 then
7  |   |   mensaje de motivacion para una nota buena;
8  |   else
9  |   |   if nota mayor que 35 y menor que 50 then
10 |   |   |   mensaje de motivacion para una nota regular;
11 |   |   else
12 |   |   |   mensaje de motivacion para una nota mala;
13 |   |   end
14 |   end
15 end
16 devolver mensaje;
```

A parte de calcular la nota del paciente 5.4, teniendo en cuenta las limitaciones surgidas durante el desarrollo de este agente, la otra función que tiene actualmente es generar el mensaje que le llega al paciente tras realizar el sistema los cálculos correspondientes. Este mensaje tiene la función de dar a conocer todos los datos que el sistema ha recopilado al paciente, entre los datos más relevantes que muestra cabe a destacar la nota, el mensaje asociado a la nota y una lista de ejercicios que el sistema recomienda al paciente que haga durante las siguientes sesiones de rehabilitación.

5.3.7. ClassificationAgent

El agente clasificador juega un rol muy importante dentro de la comunidad de agentes. Al agente clasificador se le delega la función de reconocer el movimiento que el paciente ha realizado, para que

posteriormente, el agente supervisor sea capaz de analizarlo y proporcionar una nota indicando como de bien está hecho el movimiento por el paciente. Puede que hubiera sido mejor idea hablar sobre este agente antes que sobre el agente supervisor, pero es probable que sea más sencillo entender el funcionamiento de este agente tras haber explicado previamente el funcionamiento del agente supervisor.

Para ser capaz de reconocer que movimiento ha hecho el paciente, no es posible filtrar determinadas articulaciones que intervienen en un movimiento porque evidentemente no se sabe que movimiento se va a realizar. Esto implica que hay que comparar todas y cada una de las articulaciones registradas en el sistema para cada uno de los movimientos, siendo indiferente si son relevantes durante el movimiento realizado.

Listado 5.4: Método clasificarMovimiento de Classification Agent

```
1
2  def clasificarMovimiento(self, serie, current=None):
3
4    movimientos = self.container.getMovimientosRegistrados()
5    movimientos = movimientos.split(",")
6    mov_dist = {}
7    articulaciones_por_ejes = serie.split("-,,-")
8    prxRoutine = self.broker.propertyToProxy("Exercise")
9    routineAgentPrx = ↩
         ↪ interfaz.RoutineAgentPrx.checkedCast(prxRoutine)
10   if not routineAgentPrx:
11     raise RuntimeError('Invalid proxy')
12   prxClass = self.broker.propertyToProxy("Classification")
13   clasAgentPrx = ↩
         ↪ interfaz.ClassificationAgentPrx.checkedCast(prxClass)
14   if not clasAgentPrx:
15     raise RuntimeError('Invalid proxy')
16   for k in range(0, len(movimientos)):
17     distancia = []
18     acum = 0.0
19     for i in range(0,len(articulaciones_por_ejes)):
20       serie_i = routineAgentPrx.obtenerSerie(movimientos[k], ↩
             ↪ self.articulaciones[i])
21       d = clasAgentPrx.distanciaOE(articulaciones_por_ejes[i], ↩
             ↪ serie_i)
22       distancia.append(d)
23       acum = acum + d
24     acum = acum / len(articulaciones_por_ejes)
25     mov_dist[movimientos[k]] = acum
26   distancia_minima = MAX_VALUE
27   nombre_minimo = "Not found"
28   for nombre in movimientos:
29     if mov_dist[nombre] <= distancia_minima:
30       distancia_minima = mov_dist[nombre]
31       nombre_minimo = nombre
32
33   return nombre_minimo
```

El código 5.4 hace referencia al algoritmo que determina el movimiento que ha realizado el paciente. Este método recibe como parámetro la serie que ha realizado el paciente, y la salida que produce es el nombre del movimiento que el agente ha determinado que el paciente ha realizado. La finalidad de este algoritmo es determinar en base a unas métricas que movimiento ha realizado el usuario, para ello es necesario disponer de otras series que representarán un movimiento realizado correctamente.

Lo primero es solicitar al container los diferentes movimientos que un paciente puede realizar dentro del sistema. Declaramos un vector en el que cada elemento representará la distancia entre el

movimiento de un paciente y el que se usa de referencia y recorremos los diferentes movimientos que puede realizar el usuario en un bucle. Declaramos un acumulador que representará la distancia acumulada de cada uno de los ejes que componen todas las articulaciones de un movimiento, y recorremos dentro de otro bucle todas estas articulaciones.

En este instante, el agente clasificador solicita al agente gestor de la base de datos el vector que representa el eje de una serie de un movimiento determinado, que coincide con que es el vector eje de la articulación equivalente al que se está comparando en este instante en el bucle en el que se recorren los ejes de las articulaciones del paciente. Se compara la serie del paciente con la serie representativa que aporta el sistema y se obtiene una nota, que se acumula en una variable.

Una vez finalizado el bucle se divide la nota acumulada de la comparación de todos los ejes entre el número de ejes de articulaciones que se están comparando. Esta nota es almacenada en un diccionario en el que el valor clave es el nombre del movimiento, y el valor es la distancia global respecto al movimiento que ha hecho el usuario, después de esto se realiza otra iteración del bucle hasta recorrer todos los movimientos almacenados en el sistema.

Finalmente, tras determinar las diferentes distancias respecto al movimiento realizado por el paciente, hay que determinar cual es la distancia mínima. La clave del elemento del diccionario que tenga el valor más bajo, representa el movimiento que el sistema piensa que ha realizado el paciente. Para ello simplemente se comparan las distancias y se almacena la de valor inferior.

En un primer momento puede parecer que este agente y el agente supervisor hacen lo mismo, pero hay algunos matices que los diferencian de una forma notable. Los dos utilizan el algoritmo Dynamic Time Warping para realizar la comparación de las series, pero el agente clasificador utiliza una variante de este algoritmo determinada de inicio abierto y final abierto.

La diferencia de este algoritmo respecto a la versión original, es que en la versión original se estira o se encoge la serie de prueba pasada como parámetro con el fin de buscar la similitud con la serie de referencia. En la variante de inicio y final abierto también es así, solo que en este caso, no tiene porque coincidir con el total de la serie de referencia, sino que en este caso puede solo coincidir con una subsección de la serie de referencia.

Por otro lado, el agente clasificador al no conocer el movimiento que se está realizando se ve obligado a recorrer todas las articulaciones almacenadas por el usuario, mientras que el agente supervisor, al conocer que movimiento ha realizado el usuario, solo compara las partes que requieren supervisión por parte del terapeuta y que intervienen en el movimiento.

5.3.8. MotivationalAgent

El agente motivacional tiene delegadas todas las tareas relacionadas con la finalización de un movimiento. Es un agente que en futuras actualizaciones haría que el sistema fuera más fácil de escalar. En el caso de incorporar reconocimiento facial, este sería el agente encargado de realizar la gestión entre el cliente y el agente de reconocimiento facial.

Su función podría extrapolarse a la comunicación que realiza el agente de tracking entre los clientes android y el agente clasificador y supervisor. Actualmente solo realiza la comunicación con el agente feedback debido al problema mencionado previamente de que el agente de reconocimiento facial aún no puede ser implementado.

5.4. CLIENTES ANDROID

A continuación se describirán las acciones más relevantes entre cada tipo de cliente y el servidor.

5.4.1. Cliente Paciente

Uno de los principales objetivos de este cliente es que la interacción con el sistema de cara al paciente que utilizará la aplicación sea lo más implícita posible. Para ello se han ido planteando diferentes tipos de interacción cuya implementación ha sido priorizada en función de las ventajas que proporcionaba al sistema. La idea principal para esta interacción siempre ha sido el reconocimiento por voz.

Este es un método muy eficaz a la hora de interactuar con el sistema, ya que suele tener buena acogida entre los usuarios finales. Con el objetivo de utilizarlo en este cliente, han sido estudiadas múltiples librerías de todo tipo, nativas del Unity, servicios de google e incluso alguna librería de pago. Pero todas tenían el mismo problema, no son compatibles con el dispositivo en el que se ejecuta el cliente. Finalmente se descubrió que este dispositivo no era compatible con el reconocimiento por voz por limitaciones propias del hardware, y por tanto había que buscar una alternativa.

Para resolver este problema, y evitar tener que utilizar un teclado o ratón, se decidió utilizar las manos para interactuar con el sistema. Esto fue posible gracias a la librería que permite reconocimiento de gestos. El proyecto resultante de esta implementación permite seleccionar el nombre del usuario que utilizará el sistema, y realizar transiciones entre las diferentes escenas que componen el cliente.

La comunicación entre el cliente y el servidor se realiza por medio de sockets TCP, y los datos que intercambian varían en función de la escena en la que el paciente se ubica. En la escena de inicio solo hay un flujo de comunicación. El cliente solicita al servidor los nombres de los usuarios que el terapeuta ha incluido en el sistema, para monitorizar los ejercicios de rehabilitación. Durante la escena en la que se realiza el movimiento, se realizan múltiples envíos de datos.

En este caso, el cliente envía datos al servidor y lo único que recibe por parte del servidor es un mensaje indicando que ha recibido los datos. Los datos que envía el cliente, son una instancia de todas las articulaciones que la cámara del sistema captura en un instante de tiempo. Estás articulaciones están compuestas por un sistema de coordenadas en 3D que representan una instancia de la articulación.

Uno de los comportamientos cíclicos que posee el cliente durante está escena, envía una instancia del cuerpo del paciente al servidor hasta que el cliente decide que ha terminado de realizar el movimiento. El resultado en el lado del servidor es un vector de instantes de tiempo del cuerpo del paciente. Por tanto si se recorre este vector el resultado es el movimiento del paciente, ya que se podría observar como varían las coordenadas de sus articulaciones.

Cuando el paciente decide que ha finalizado el movimiento, pasaría a la escena de análisis. Durante esta escena, el cliente realiza una petición al servidor solicitando los resultados del último movimiento que el paciente ha realizado. El servidor responde con una serie de datos que son interpretados en el lado del cliente para ser mostrados al paciente. Estos datos son mostrados al paciente durante el tiempo que el considere necesario. Finalmente, cuando el paciente realice un gesto, se vuelve a la pantalla de inicio donde empieza de nuevo todo el proceso descrito.

5.4.2. Cliente Administrador

La función de este cliente es incorporar nuevos movimientos, nuevos usuarios y comprobar el correcto funcionamiento en el cliente que utilizará el paciente. En la pantalla de inicio se pueden observar varios elementos de la interfaz. A diferencia de la interfaz del cliente que utilizará el paciente, en esta no se utiliza el reconocimiento de gestos para interactuar con el sistema, salvo para finalizar un movimiento.

Hay un selector con los diferentes movimientos que hay disponibles en el sistema. Este selector se utiliza exclusivamente cuando se incorpore un movimiento en el sistema, o cuando se pretenda sobreescribir uno ya existente. Por otro lado, hay una entrada de texto, donde se debe escribir el nombre de un usuario. Si este usuario ya existe en el sistema, simplemente se añadirá un movimiento nuevo a su historial de movimientos realizados.

Por otro lado, si el usuario no existe dentro del sistema, a parte de la acción mencionada anteriormente, también se registrará este usuario para que este disponible en la aplicación que utilizará el paciente. A parte, en la interfaz de inicio hay dos botones. El botón de añadir movimiento y el botón de analizar movimiento.

- **Añadir movimiento:** Este botón dirige a la escena donde se realiza un movimiento en el sistema. Envía datos iterativamente al servidor con las coordenadas tridimensionales de las articulaciones que está reconociendo del usuario y el movimiento que se está realizando. Una vez finalizado el movimiento, el cliente envía una señal al servidor indicando que se está añadiendo un movimiento para que sea utilizado como referencia. Llegado a este punto, la aplicación cliente vuelve a la escena de inicio.

- **Analizar movimiento:** Este botón conduce a la escena de análisis de un movimiento. Está escena es una réplica de la función de análisis que realiza el cliente paciente para corroborar su correcto funcionamiento. La única diferencia es que el nombre del usuario puede ser indicado libremente para poder incorporar nuevos usuarios en el sistema. A diferencia de la escena descrita anteriormente, tras realizar el movimiento en esta escena se transita a la escena del análisis del movimiento. En esta escena se indica el movimiento que el usuario ha realizado y la nota que ha obtenido. Tras ello, se vuelve a la escena de inicio de la aplicación cliente.

5.5. INTERFAZ WEB

Como se ha mencionado en la introducción, es necesario algún medio de comunicación entre los datos que proporcionan los pacientes al sistema y el terapeuta. Utilizar una interfaz web tiene muchas ventajas. Entre ellas podemos destacar que no es necesario tenerla instalada en un determinado equipo, permitiendo al terapeuta acceder desde cualquier lugar a la interfaz. No dependen del sistema operativo que posee el equipo desde el que se consulta el estado del paciente. Tampoco requiere realizar una instalación y es posible actualizarla de manera remota.

Dejando de lado las ventajas que posee una aplicación web, el uso de una interfaz web ha supuesto algunos inconvenientes. Para que el tiempo de respuesta sea bajo es necesario disponer de una buena conexión de red, y esto implica que requiere conexión al exterior (a no ser que se decida desplegar en una red local). Por otra parte, lo ideal habría sido que la API que utiliza la interfaz para recuperar datos del sistema hubiera estado integrada dentro del sistema de ZeroC Ice.

El problema es que esto no es posible debido a que ZeroC Ice no es compatible con la tecnología que utiliza la interfaz web, en este caso Flask, un framework de Python. Era posible incluir el script que ejecuta la API como nodo del servidor, pero este no permitía hacer una petición desde una herramienta como Postman o un cliente web. Por tanto, en vez de ser un agente más del sistema y hacer peticiones al RoutineAgent para obtener los datos, ha sido necesaria una conexión independiente a la de este agente con la base de datos.

Para la implementación de la interfaz ha sido necesario un sistema de inicio de sesión por motivos de seguridad. Una vez iniciada sesión con los credenciales del terapeuta, se proporciona al navegador del terapeuta un token que deberá incluir en todas las peticiones que haga a la API. Salvo a los credenciales de inicio de sesión de otros usuarios, el usuario registrado tiene acceso a todos los datos de los pacientes.

Estos datos pueden ser visualizados en forma de lista, en cuyo caso se muestra el nombre del usuario que ha realizado el movimiento, el movimiento que ha realizado, y la nota que ha obtenido. También es posible visualizarlos en un gráfico, permitiendo simplificar la comprensión de la evolución que está teniendo efecto en el paciente.

5.6. ORGANIZACIÓN Y COMPONENTES DEL PROYECTO

A continuación se mostrará la estructura del proyecto con el objetivo de tener una visión global. El sistema está compuesto por cuatro prototipos de las aplicaciones que manipulan el conjunto de los datos del sistema.

5.6.1. Estructura general del cliente paciente

CLIENTE PACIENTE	/			
	Assets/	**Animation/**	**MainMenu/**	Closed.anim
				MainMenu.controller
				Open.anim
			Panel/	Ficheros de animación
			Settings/	Archivos de configuración
			Rotation.admin	
			SF Button.controller	
		CompareMovimiento/	EscenaCompararUnity	
		Fonts/	Diferentes fuentes utilizadas en la aplicación.	
		Materials/	Manteriales utilizados para las interfaces	
		Menu/	DropdownScript.cs	
			levelManager.cs	
			Main.unity	
			titulo.png	
		NuitrackSDK/	Kit de desarrollo de aplicaciones con Nuitrack	
		Plugins/	Archivos de configuración Android	
		addMovScript.cs		
		CustomPointer.cs		
		datos_scenas_script.cs		
		estadoMano.cs		
		feedback.cs		
		FeedBack.unity		
		feedmanager.cs		
		informacion.cs		
		SkeletonControllerCustom.cs		
		unityhospital.png		
	Library/			
	Logs/			
	Packages/			
	ProyectSettings/			
	UnityPackageManager/	Archivos de Configuración de Unity		
	Assembly-CSharp-csproj			
	Cliente.csproj			
	Cliente.Editor.csproj			
	Cliente.sln			

Figura 5.6: Estructura general del cliente paciente

La figura 5.6 representa la estructura y los componentes de la aplicación que utilizará el paciente en el sistema. Es una estructura similar a la de cualquier proyecto diseñado con Unity. Dentro del directorio Assets/ se localizan todos los componentes que se han desarrollado para la aplicación. Entre los componentes más relevantes de este directorio podemos destacar la escena en la que se realiza la comparación junto a todos los scripts que utiliza. Las fuentes utilizadas dentro del sistema para dar

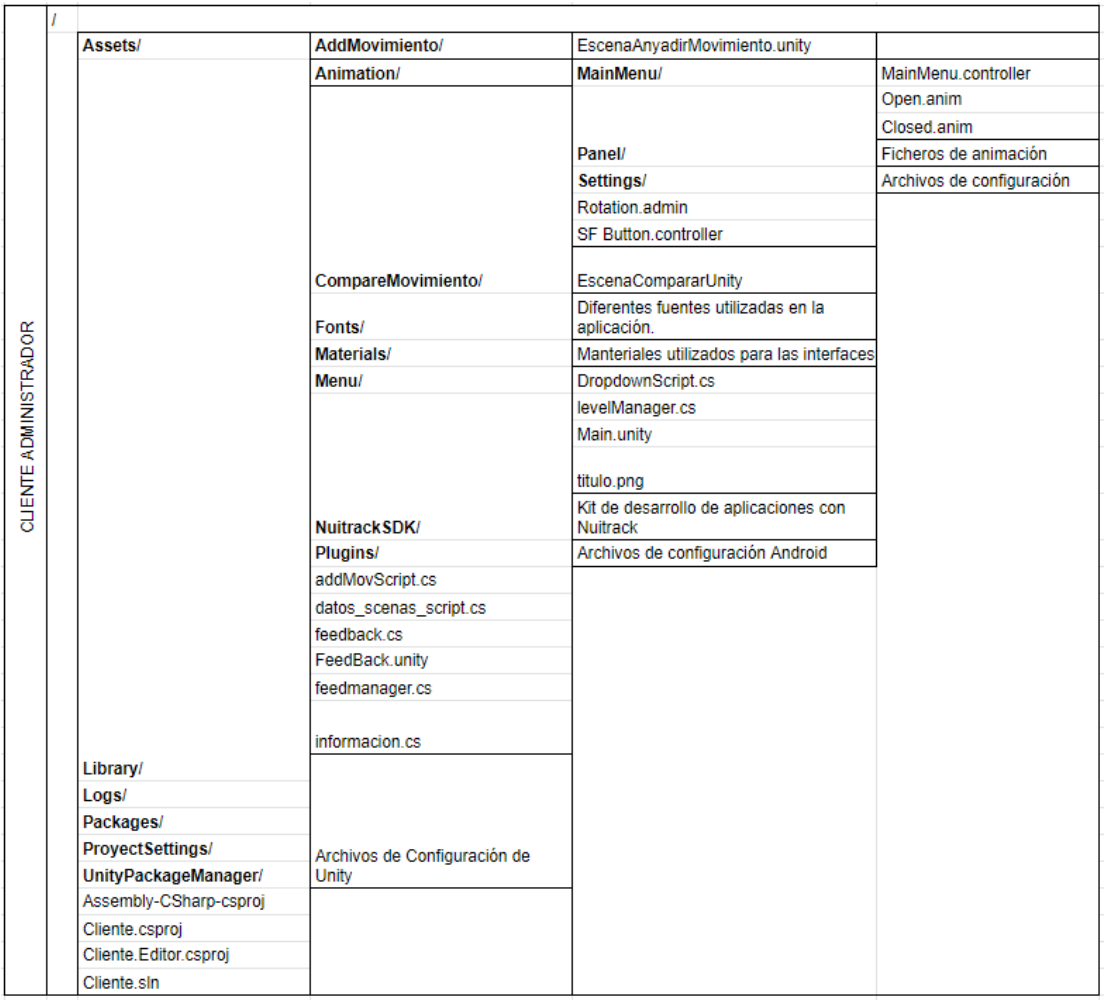

Figura 5.7: Estructura general del cliente administrador

formato a los textos de las diferentes interfaces. El paquete de Nuitrack que permite establecer una comunicación con la aplicación. La escena de inicio junto a sus componentes y la escena de análisis.

Volviendo al directorio principal del proyecto tras describir el directorio assets/. El resto de directorios son generados por Unity y contienen principalmente la configuración del proyecto. Se puede dar más relevancia al directorio library/ que contiene las librerías utilizadas dentro del cliente.

5.6.2. Estructura general del cliente administrador

La estructura del cliente administrador es similar a la del cliente paciente. Esto es debido a que ambos están desarrollados en Unity y comparten parte de la funcionalidad con variaciones. En este caso, dentro del directorio assets/ se puede apreciar otro directorio denominado AddMovimiento/. En el que se encuentra la escena en la que se incorporan o actualizan movimientos en la base de datos del servidor. Por otro lado, tiene menos elementos estéticos, ya que no debe ser tan atractiva como la que utilizará el paciente final.

5.6.3. Estructura general del administrador web

El proyecto asociado a toda la parte de administrador web queda reflejado en la figura 5.8. Se queda dividido en dos partes, por un lado el fichero del directorio raíz. Este fichero llamado aplicacion.py representa el backend de la aplicación. Es utilizado como API para hacer llamadas desde el cliente

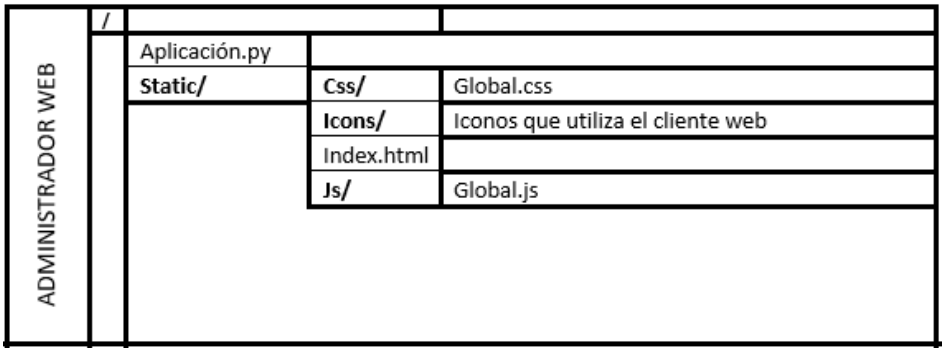

Figura 5.8: Estructura general del servidor web

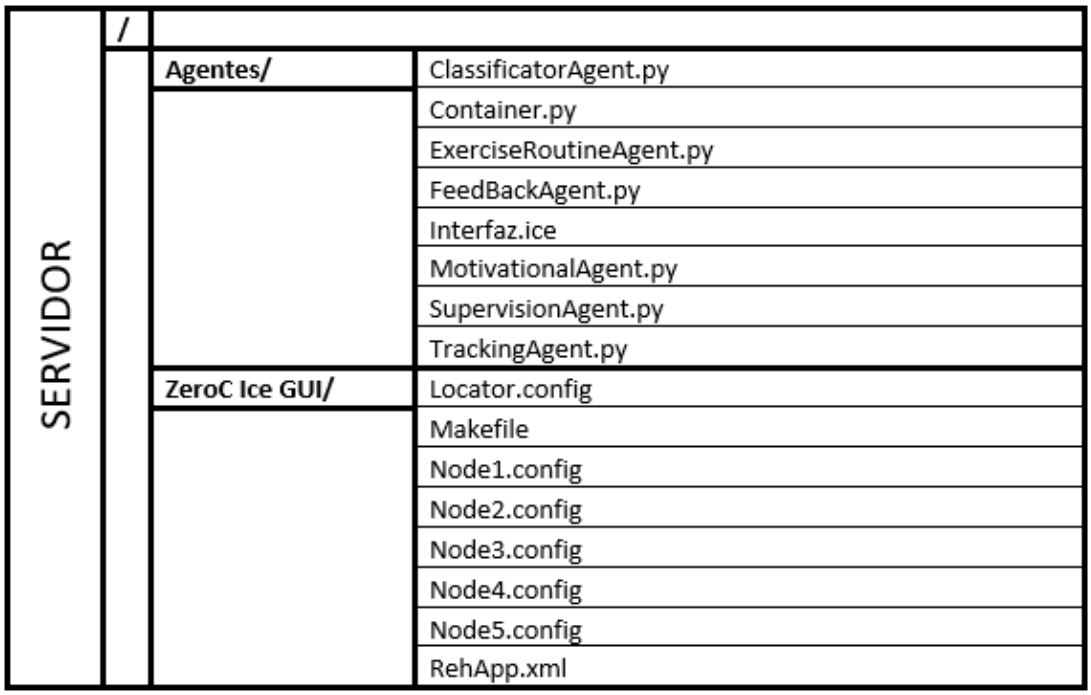

Figura 5.9: Arbol de directorios del servidor

y recuperar información de la base de datos. Por otro lado, todos los ficheros del directorio static/ representan el frontend de la aplicación. Dentro del directorio css/ se encuentran las hojas de estilos y dentro de la carpeta js/ los scripts y librerías que componen este proyecto. También hay un directorio con fuentes de iconos externas y finalmente, el fichero index.html. Este fichero contiene toda la estructura en código html a la que luego se le aplican los estilos y scripts.

5.6.4. Estructura general del servidor

La distribución de los ficheros que componen el servidor es mostrada en la figura 5.9.

El directorio **agentes** contiene tanto el servidor como cliente web del panel de monitorización del terapeuta, la base de datos y los ficheros de los agentes que componen el sistema. Esto es así porque en un primer momento lo ideal habría sido que la parte de administración web hubiera formado parte del sistema de Zero C Ice, pero esto no ha sido posible debido a una incompatibilidad entre el framework que despliega la aplicación web con Zero C Ice de la que hablaremos adelante.

En el directorio **ZeroC ICE GUI** se encuentran todos los archivos de configuración de los nodos del sistema, el fichero xml que carga los datos en la interfaz gráfica de Zero C Ice y un Makefile que es

utilizado para arrancar el sistema. El Makefile no solo arranca el sistema de Zero C Ice. También sirve para poner en marcha el servidor Rserve y para migrar todos los ficheros a una carpeta temporal de ubuntu.

5.7. HISTORIAS DE USUARIO

En esta sección se muestra el conjunto final de historias de usuario.

#01	Como **cliente** quiero **un sistema** para poder **monitorizar ejercicios de rehabilitación**	
CRITERIOS DE ACEPTACIÓN		
1.	**Interacción gestual**	En caso que **el paciente utilice la aplicación**, cuando **decida realizar un movimiento**, el sistema **no debe requerir uso de teclado o ratón.**
2.	**Incoporar movimientos**	En caso que **el terapeuta quiera incorporar un movimiento,** cuando **lo considere oportuno**, el sistema **debe de proporcionar una parte de administración para él.**
3.	**Reconocer movimiento**	En caso que **el paciente realice un movimiento**, cuando **tenga que realizar los ejercicios de rehabilitación,** el sistema **debe reconocer el movimiento que ha realizado sin que el paciente lo indique explícitamente.**
4.	**Monitorizar movimiento**	En caso que **el terapeuta quiera monitorizar ejercicios de un paciente**, cuando **corresponda hacer un seguimiento al paciente**, el sistema **debe proporcionar una interfaz desde la que visualizar los resultados.**

Figura 5.10: Historia de usuario - 1

#02	Como **cliente** quiero **una aplicación para el paciente** para poder **realizar los ejercicios de rehabilitación**	
CRITERIOS DE ACEPTACIÓN		
1.	**Reconocimiento del cuerpo paciente**	En caso que **el paciente se ubique delante de la cámara**, cuando **pretenda realizar un ejercicio de rehabilitación**, el sistema **debe reconocer el cuerpo del paciente**
2.	**Reconocimiento de gestos con las manos**	En caso que **el paciente pretenda interactuar con la interfaz**, cuando **decida realizar un movimiento**, el sistema **debe reconocer que gesto está haciendo para transitar entre las escenas**
3.	**Identificación en el sistema**	En caso que **el paciente decida quiera usar el sistema**, cuando **realice los ejercicios de rehabilitación**, el sistema **debe ser capaz de reconocer que usuario va a realizar el movimiento.**
4.	**Calificar un movimiento**	En caso que **el paciente este delante de la cámara**, cuando **finalice un ejercicio**, el sistema **debe calificar el ejercicio con una nota que exprese como de bien hecho está el ejercicio.**

Figura 5.11: Historia de usuario - 2

#03	Como **colaborador** quiero **clasificar ejercicios** para poder **evaluar la nota de los ejercicios**	
CRITERIOS DE ACEPTACIÓN		
1.	**Debe diferenciarse**	En caso que **el usuario realice el movimiento**, el sistema **debe ser capaz de diferenciarlo de los demás ejercicio.**
2.	**Obtener Calificación**	En caso que **el ejercicio se haya realizado exitosamente**, el sistema **debe ser capaz de proporcionar una nota.**

Figura 5.12: Historia de usuario - 3

#04	Como **terapeuta** quiero **añadir nuevos usuarios** para poder **monitorizar su rehabilitación.**	
CRITERIOS DE ACEPTACIÓN		
1.	**Un único nombre por usuario**	En caso que **el paciente inicie sesión**, el sistema **debe ser capaz de diferenciar a cada usuario.**
2.	**No requiera usar un teclado**	En caso que **el paciente elija su usuario**, cuando **pretenda realizar un movimiento**, el sistema **no le solicitará el uso de un teclado para acceder con su usuario.**

Figura 5.13: Historia de usuario - 4

#05	Como **terapeuta** quiero **registrar la nota de los pacientes** para poder **obtener estadísticas.**	
CRITERIOS DE ACEPTACIÓN		
1.	**Represente la realidad**	En caso que **el paciente finalice un ejercicio**, el sistema **proporcionará una nota que represente si el ejercicio ha sido realizado realmente bien o no.**
2.	**Debe ser almacenada**	En caso que **el paciente finalice un ejercicio**, el sistema **debe ser capaz de almacenar la nota**
3.	**Debe mostrar la nota al usuario**	En caso que **el ejercicio**, el sistema **mostrará la nota al usuario en tiempo real.**

Figura 5.14: Historia de usuario - 5

#06	Como **terapeuta** quiero **obtener estadísticas** para poder **monitorizar la evolución de cada paciente**	
CRITERIOS DE ACEPTACIÓN		
1.	**Representar la evolución a través de gráficos**	En caso que **el terapeuta revise la evolución de un paciente**, cuando **tenga que realizar su evaluación**, el sistema **mostrará su evolución de forma gráfica**
2.	**Mostrar evolución general**	En caso que **el terapeuta revise la evolución de un paciente**, cuando **tenga que realizar su evaluación**, el sistema **permitirá visualizar los resultados en conjunto de todos los movimientos**
3.	**Mostrar evolución específica**	En caso que **el terapeuta revise la evolución de un paciente**, cuando **tenga que realizar su evaluación**, el sistema **permitirá visualizar los resultados de un determinado movimiento**

Figura 5.15: Historia de usuario - 6

#07	Como **colaborador** quiero **registrar movimientos en el sistema** para poder **clasificarlos**	
CRITERIOS DE ACEPTACIÓN		
1.	**Requiere un identificador**	En caso que **se incorpore un movimiento al sistema**, cuando **el terapeuta cree un nuevo movimiento**, el sistema **debe almacenar el movimiento con un identificador**
2.	**Requiere el nombre del movimiento**	En caso que **se incorpore un movimiento al sistema**, cuando **el terapeuta cree un nuevo movimiento**, el sistema **debe almacenar el movimiento con un nombre que indicará al paciente que movimiento ha realizado**
3.	**Almacenar todos las articulaciones**	En caso que **se incorpore un movimiento al sistema**, cuando **el terapeuta cree un nuevo movimiento**, el sistema **debe registrar todas las articulaciones que componen el movimiento del paciente, independientemente de si son relevantes durante el movimiento**

Figura 5.16: Historia de usuario - 7

#08	Como **paciente** quiero **obtener una lista de ejercicios** para poder **realizarlos cuando me corresponda**	
CRITERIOS DE ACEPTACIÓN		
1.	**Proporcionarse en tiempo real**	En caso que **el paciente finalice su movimiento,** cuando **se le proporcione una nota**, el sistema **debe recomendarle una serie de ejercicios adicionales para que los haga posteriormente.**

Figura 5.17: Historia de usuario - 8

#09	Como **cliente** quiero **un sistema escalable** para poder **ampliar la funcionalidad en otra iteración.**	
CRITERIOS DE ACEPTACIÓN		
1.	**Framework escalable**	El sistema debe permitir ampliar su funcionalidad en un futuro, ya sea a través de mejoras o incorporando nuevas funciones.

Figura 5.18: Historia de usuario - 9

CAPÍTULO 6

RESULTADOS

6.1. ASPECTO Y FUNCIONALIDAD DEL PROTOTIPO FINAL

Como ya se ha explicado previamente, el prototipo propuesto en este proyecto está compuesto por cuatro elementos. El servidor web que utiliza el administrador del sistema para revisar la evolución de los pacientes. El servidor encargado de realizar la clasificación y el análisis de los movimientos proporcionados por el paciente, o de incorporar nuevos movimientos estipulados por el terapeuta. Por otro lado el cliente Android que sería usado por los pacientes finales para realizar ejercicios de rehabilitación. Y por último, el cliente que utilizaría el terapeuta para incorporar o editar los movimientos del sistema.

6.1.1. Cliente del paciente

Este determinado cliente dispone de una sencilla interfaz que no necesita hacer uso de ningún dispositivo diferente a la cámara, ya que se maneja con los gestos de las manos. Cuando ejecutas la aplicación, lo primero que observamos en la figura 6.1 es un menú en el que debemos seleccionar haciendo uso de gestos con las manos uno de los nombres de pacientes listados en la interfaz.

Figura 6.1: Cliente Android Rehabilitación Remota en la escena inicial

Navegar por la aplicación haciendo uso de los gestos es muy sencillo. Lo primero que hay que hacer es ubicarte delante de la cámara. Una vez que estás colocado verás que aparecen dos manos al igual que en la figura 6.2.

Figura 6.2: Cliente Android Rehabilitación Remota en la escena inicial tras reconocer las manos

Esto es síntoma de que el sistema ya ha reconocido las manos del usuario y de que ya está listo para realizar cualquier gesto. Ahora es el momento de elegir el nombre del usuario que va a realizar el movimiento. Para seleccionarlo es necesario realizar determinados gestos. Como se puede observar, hay una lista de nombres justo debajo del título de la aplicación. El nombre situado en el centro es el nombre seleccionado actualmente. A continuación se explicará como transitar en este selector utilizando gestos.

Si el usuario cierra la mano, la mano situada a la derecha y el texto que hay debajo cambiarán por un puño cerrado y LISTO. Esto es síntoma de que el usuario ha cerrado el puño derecho, pero tiene el izquierdo abierto. La reacción del sistema ante este evento será que el selector para elegir el nombre del usuario transitará a la derecha.

Del mismo modo que si cierras el puño derecho con la mano izquierda abierta, gira a la derecha, si cierras el puño izquierdo con la mano derecha abierta, gira a la izquierda.

Figura 6.3: Cliente Android Rehabilitación Remota en la escena inicial antes de transitar a la fase de análisis

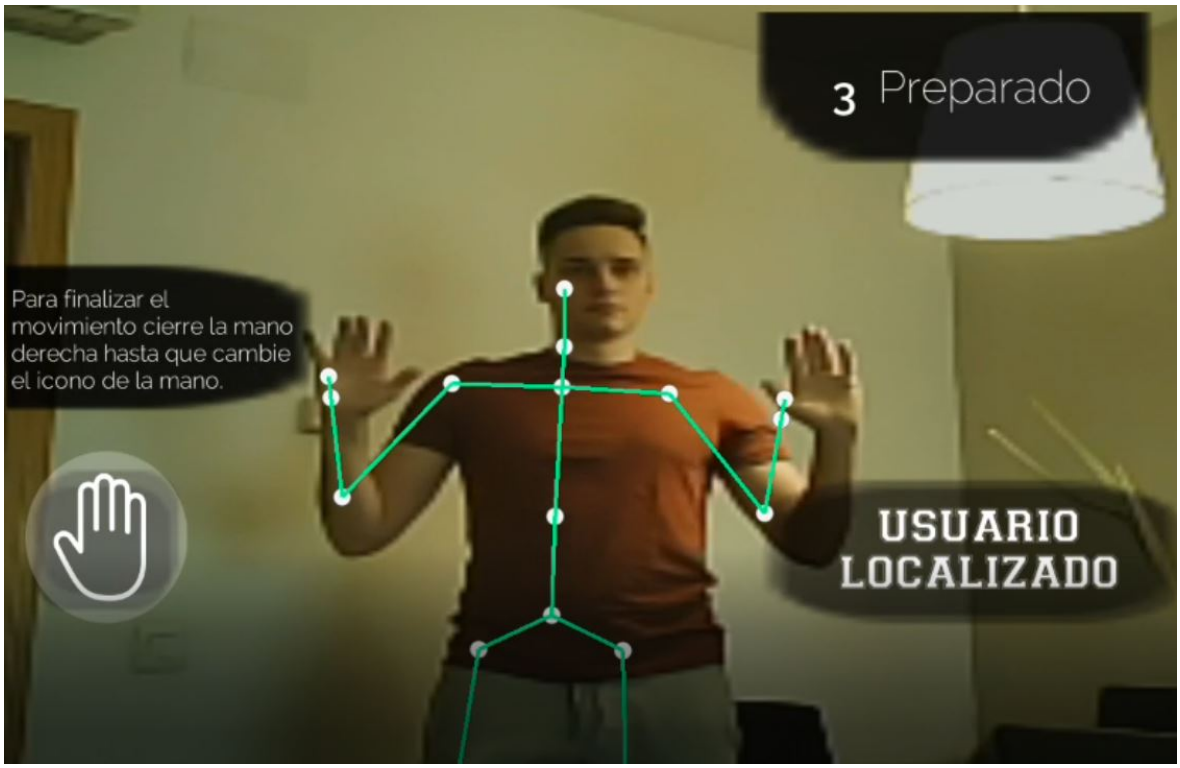

Figura 6.4: Cliente Android Rehabilitación Remota en la escena de análisis

Ya se ha explicado el efecto que tiene en el sistema cerrar una mano u otra, pero siempre teniendo la opuesta abierta. Cuando se cierran las dos simultáneamente, el sistema queda tal como en la figura 6.3. Pocos segundos después el sistema transita a la escena de análisis.

En la figura 6.4 se puede observar como es la escena durante la que el sistema captura el movimiento del usuario. En ella se pueden apreciar diferentes mensajes que permitan al usuario comprender que está pasando en el sistema. Por un lado, vuelve a aparecer la figura de una mano cuya función es indicar al usuario si reconoce que su mano está abierta o cerrada. Esto es importante debido a que una vez iniciada la captura del movimiento, si el usuario cierra el puño le indica al sistema que ha acabado de hacer el movimiento.

Justo encima de la mano que se acaba de comentar, se visualiza un mensaje indicando la función de esta mano. Justamente encima de la posición del usuario que realiza el movimiento, se ubica una serie de puntos unidas con líneas rectas que indican lo que la cámara interpreta respecto a lo que captura del cuerpo del usuario. En la parte derecha aparece un mensaje indicando si el sistema está recuperando el esqueleto del usuario o no. Por otro lado, en la parte superior derecha aparece un número y un mensaje.

Este mensaje varía en función del número que aparezca en el contador, como se puede apreciar el cambio entre la figura 6.4 y la figura 6.5. Esto es debido a que el usuario al entrar en escena, necesita un tiempo para prepararse para realizar el movimiento. Por tanto se le proporcionan cinco segundos para colocarse correctamente delante de la cámara.

Como se puede ver cuando el contador llega a cero, el sistema se prepara para capturar el movimiento del usuario. Este movimiento puede ser tan largo como el usuario desee, y cuando decida que el movimiento ha acabado, lo único que debe hacer es cerrar el puño para que el sistema transite a la escena de análisis.

Tras cerrar la mano por parte del usuario cuando realiza el movimiento en la fase de análisis, el sistema transita a la fase en la que se le proporciona al usuario información. Está información le indica al usuario como de bien hecho está el movimiento, le proporciona una nota y le da una

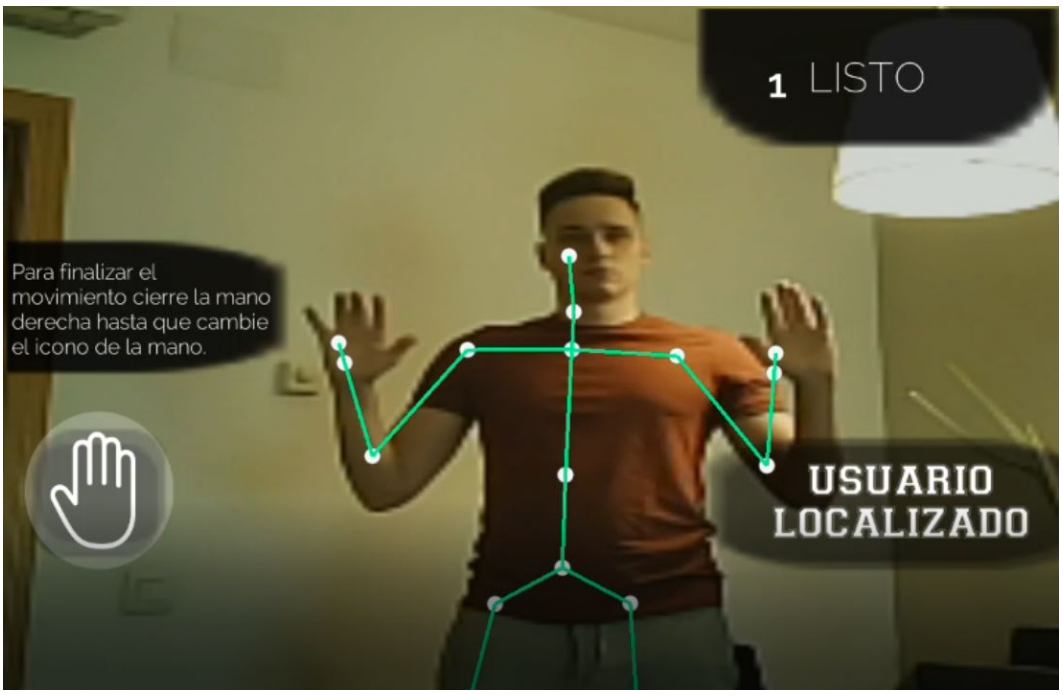

Figura 6.5: Cliente Android Rehabilitación Remota en la escena de análisis justo antes de iniciar el movimiento

serie de recomendaciones indicadas por el terapeuta para indicar posteriores ejercicios que podría hacer. En la figura 6.7 se puede ver un ejemplo del resultado del análisis de un movimiento. En el mostrado particularmente, se puede ver que el usuario A ha realizado un movimiento levantando ambos brazos, y que ha obtenido una nota de 116 puntos.

A diferencia de lo que se pueda pensar en un primer momento, cuanto más alta es la nota, peor realizado está el ejercicio. Por tanto, lo ideal sería que la nota fuera 0, para indicar que el ejercicio ha salido perfecto. Por otro lado, está nota se transforma en el mensaje que aparece en conclusiones, que como se ha indicado, es una nota bastante mala. También se le ofrece una lista con los ejercicios que el usuario debería realizar para seguir con su rehabilitación.

Figura 6.7: Cliente Android Rehabilitación Remota en la escena de feedback

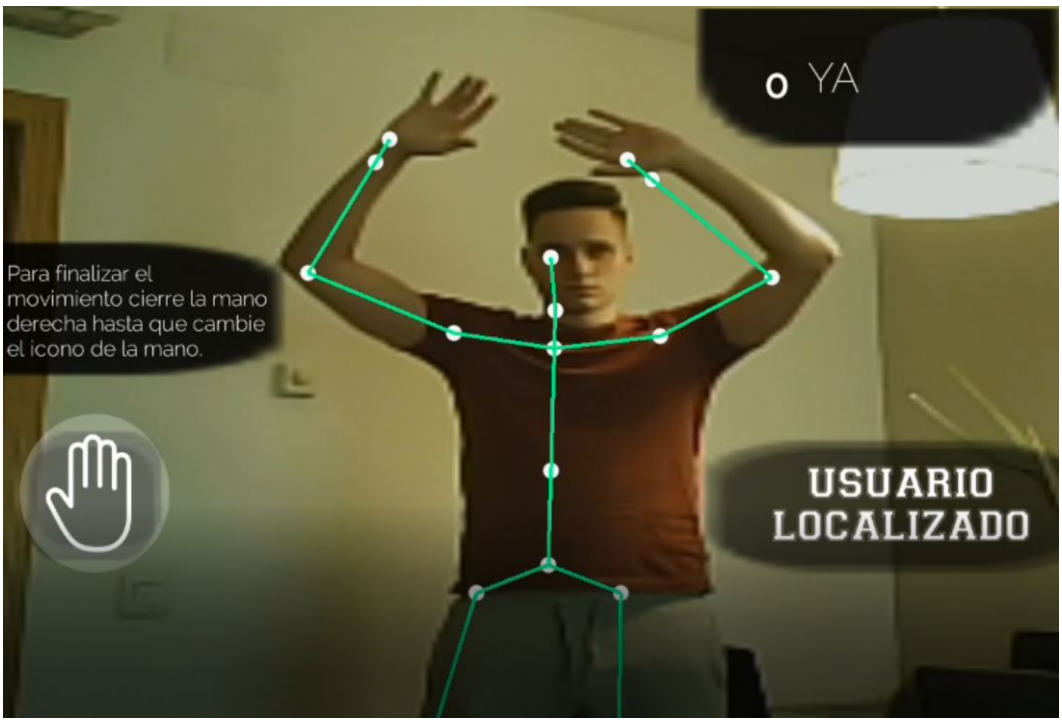

Figura 6.6: Cliente Android Rehabilitación Remota en la escena de análisis durante la captura del movimiento

6.1.2. Cliente del administrador

Figura 6.8: Cliente Administrador en la escena inicial

A continuación se explicarán los resultados del cliente administrador. Como se puede observar en la figura 6.8, la interfaz es muy similar a la del cliente que utilizarán los paciente. Se pueden destacar varias diferencias. Por un lado, está interfaz cuenta con dos botones, uno para incorporar un movimiento nuevo al sistema o actualizar uno existente. El otro botón es utilizado para realizar pruebas por parte del administrador del sistema. Este botón tiene la misma funcionalidad que la interfaz del paciente, por lo que el administrador puede monitorizar si el ejercicio se ha integrado adecuadamente en el sistema.

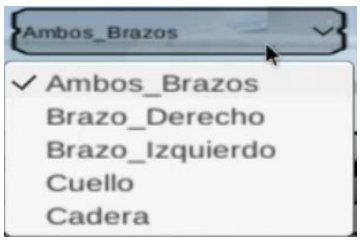

Figura 6.9: Cliente Administrador en el menú desplegable de la escena inicial

Por otro lado, también cuenta con un menú desplegable mostrado en la figura 6.9. Se utiliza solo al pulsar el botón de añadir movimiento. Para cargar los datos que se visualizan en la figura es necesario realizar una petición al servidor y es este el que retorna los datos que se cargarán en el selector.

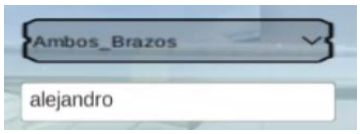

Figura 6.10: Cliente Administrador en el campo de texto de la escena inicial

Y para finalizar la descripción de esta escena, como se puede observar en la figura 6.10 también dispone de un campo de texto situado justamente debajo del selector desplegable. En este campo de texto se introduce el nombre del usuario que va a realizar el movimiento. Este campo de texto en la interfaz del paciente era un selector desplegable. Si en esta escena también fuera un menú desplegable no se podrían incorporar nuevos pacientes al sistema. Si el paciente que se va a incorporar ya existe, no se añade uno nuevo, simplemente se añade un nuevo registro para su evaluación.

Después se llegaría a la pantalla en la que se vería el resultado de pulsar el botón de añadir o analizar movimiento, tanto uno como otro llevan a esta escena. Esto es normal debido a que tanto en un caso como en otro es necesario capturar un movimiento. Llegado a este punto es necesario que el usuario se ponga delante de la cámara. En este prototipo se proporcionan cinco segundos para colocarte, aun que este parámetro se podría modificar sin ningún problema.

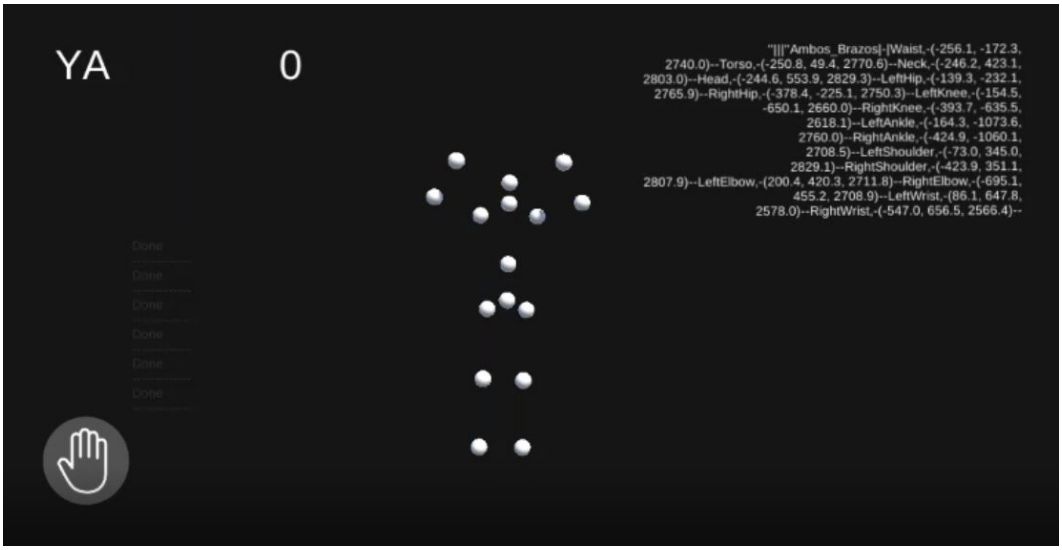

Figura 6.11: Cliente Administrador en la escena de análisis y añadir movimiento tras detectar al usuario

Después de que el usuario se coloque delante de la cámara y el contador llegué a cero comenzará

la captura del movimiento por parte del sistema. En la figura 6.11 se puede ver un ejemplo del estado de la interfaz en este momento. Al igual que en el anterior cliente, el movimiento también finaliza al cerrar una mano, a pesar de que el resto de la interfaz se maneje haciendo uso de periféricos.

Tras realizar el movimiento y por consecuencia, que el usuario cierre la mano, transitaremos a una escena u otra en función del botón que haya sido pulsado en la escena de inicio.

En el caso de que este siendo analizado un movimiento, el sistema transitará a la escena de resultados para mostrar los datos que el sistema ha procesado. Si por el contrario, hemos incorporado un movimiento, el sistema transitará a la escena de inicio ya que no se ha hecho ninguna comparación.

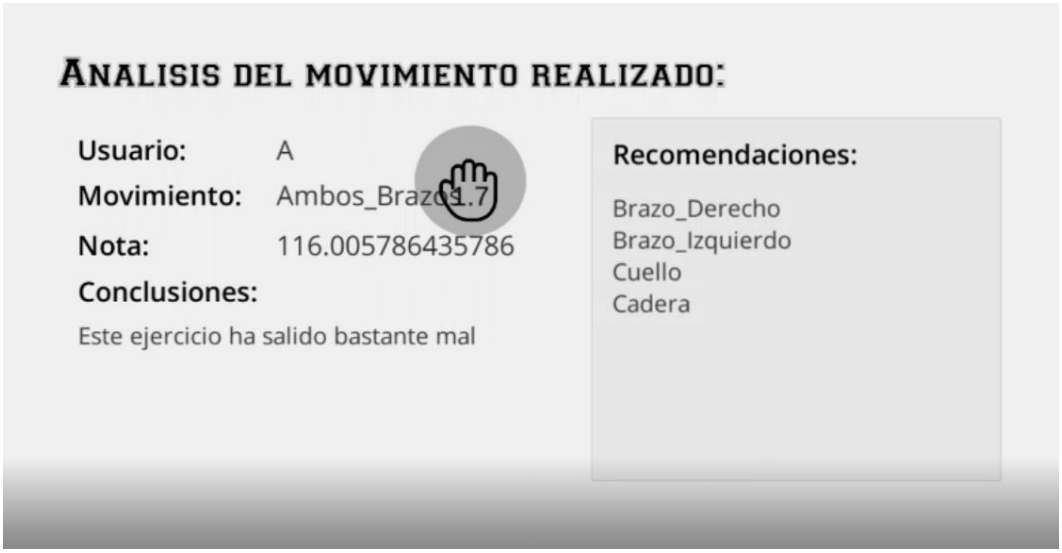

Figura 6.12: Cliente Administrador en la escena de resultados

Finalmente, en la figura 6.12 podemos observar la escena de resultados. Como está escena es exactamente igual que la escena visualizada en el cliente, no es necesario comentarla.

6.1.3. Servidor

Respecto al servidor en el capítulo de 5 se ha hecho un análisis de como funcionan cada uno de los agentes. Se encarga de proporcionar las respuestas a los clientes que componen el sistema. Puesto en funcionamiento tiene una interfaz para administrar los nodos del sistema. Está interfaz se puede ver en la figura 6.13.

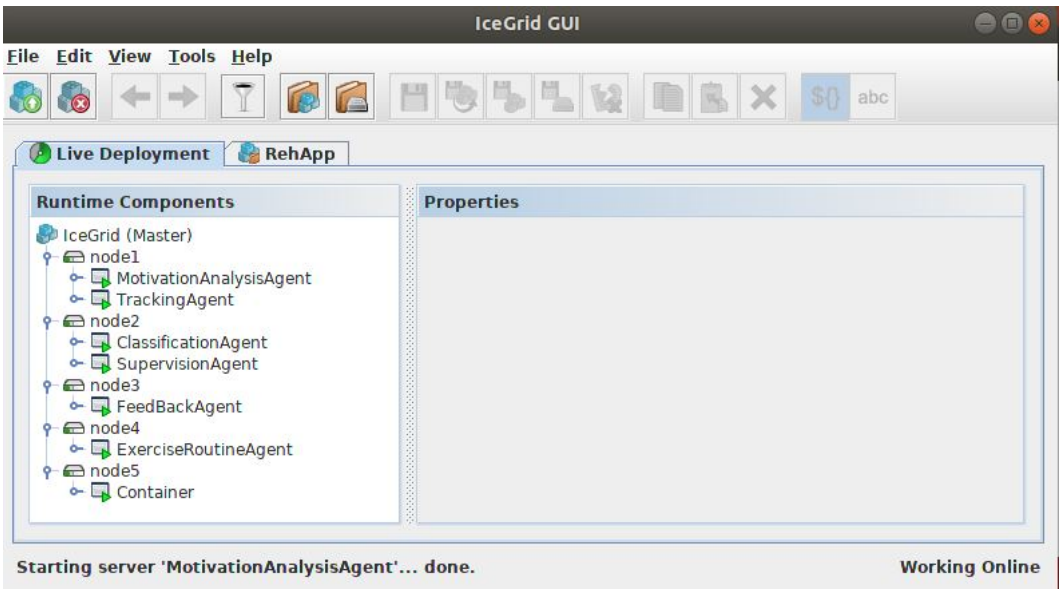

Figura 6.13: Servidor del sistema

Desde esta interfaz se administran las salidas de consola y de errores de los diferentes servidores que componen los nodos del sistema. También es posible configurar propiedades para que en el caso de que se incorpore un nuevo nodo, los demás agentes lo puedan identificar.

A continuación se mostrarán una serie de casos mostrando los tiempos de ejecución del servidor en diferentes iteraciones de uso.

En la figura 6.14 se pueden observar los tiempos resultantes de realizar una consulta para obtener la lista de movimientos disponibles en el sistema. Está lista se utiliza en el cliente administrador para incorporar nuevos movimientos al sistema. Las pruebas se han realizado con cinco movimientos diferentes en el sistema.

Tiempos de consultar movimientos disponibles							
Iteración	1	2	3	4	5	6	7
Resultado	0,000367641	0,000257254	0,000414371	0,000280142	0,000247002	0,000282526	0,000254393

Figura 6.14: Tiempo necesario para consultar los movimientos disponibles en el sistema

En la figura 6.15 se muestran los tiempos resultantes de obtener los usuarios disponibles en el sistema. Está consulta es realizada por el cliente que usa el paciente para escoger su usuario.

Tiempos de consultar usuarios disponibles							
Iteración	1	2	3	4	5	6	7
Resultado	0,000564371	0,000467642	0,000239002	0,00065015	0,000164872	0,000309226	0,000239025

Figura 6.15: Tiempo necesario para consultar los usuarios disponibles en el sistema

Tiempos de comparación y analisis de un movimiento							
Captura del movimiento	0,0500082970	1,1377222538	5,1696860790	7,0093581676	12,5743362904	16,2895927429	17,0081114864
Clasificar movimiento	0,6678173542	0,7266361713	3,0244975090	3,0508251190	5,0480806828	8,0673084259	5,1431357861
Evaluar movimiento	0,2098350525	0,1236491203	0,0760588646	0,0807287693	0,1569066048	0,2599766254	0,8537461758
Escribir en base de datos	0,0039284229	0,0101313591	0,0106720924	0,0130794048	0,0107910633	0,0103378296	0,0102450848

Figura 6.16: Tiempo necesario para clasificar y analizar un movimiento

Por otro lado, en la figura 6.16 se muestran los tiempos de clasificar, evaluar y escribir el resultado en base de datos de un movimiento realizado por el paciente. Los registros están ordenador de menor

a mayor tiempo de captura del movimiento. Este tiempo hace referencia al tiempo que un usuario tarda en realizar un movimiento completo.

Como se puede observar los tiempos de clasificación y de evaluación del movimiento crecen linealmente en función del tiempo que el usuario tarda en realizar el movimiento. En cambio si se incorporarán nuevos movimientos al sistema y compararamos los tiempos con los actuales, se podría observar que los tiempos crecen exponencialmente.

Tiempos para añadir un movimiento							
Captura del movimiento	0,0713222027	6,2630674839	6,6409547329	13,0058727264	13,0841059685	16,6953549385	31,6985459328
Escribir en base de datos	0,0138792992	0,0038578510	0,0059010983	0,0178611279	0,0119421482	0,0155966282	0,0073144436

Figura 6.17: Tiempo necesario para añadir un nuevo movimiento

Finalmente en la figura 6.17 se pueden apreciar los tiempos de ejecución de incorporar un nuevo movimiento a la base de datos. En términos generales se puede apreciar que la tendencia es a mayor tiempo de captura del movimiento, mayor es el tiempo en escribir en base de datos.

6.1.4. Administrador web

Con el objetivo de realizar un seguimiento a los pacientes que utilizan el sistema, se ha implementado una interfaz web que utilizará el terapeuta. Para proporcionar seguridad al sistema, se implementó un sistema de inicio de sesión que impida que cualquier usuario acceda al sistema.

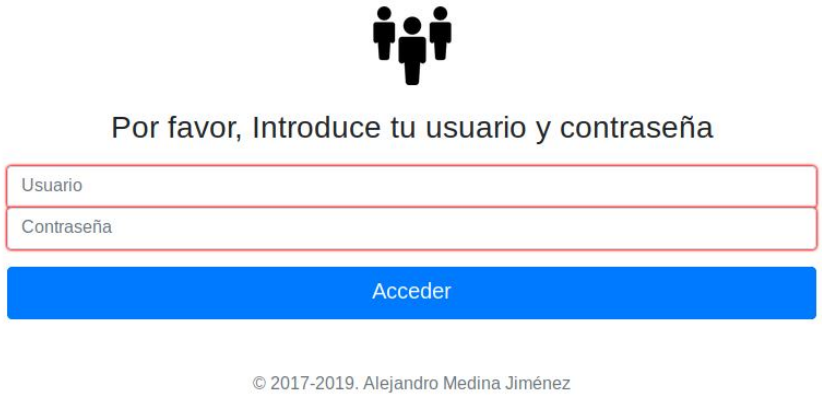

Figura 6.18: Administrador Web en la ventana de inicio de sesión

En la figura 6.18 se puede observar la interfaz de inicio de sesión. Al igual que todo el tema, utiliza bootstrap para facilitar el diseño de la interfaz y devuelve un token, que el cliente proporcionará en cada petición al sistema, para indicar que es un usuario registrado.

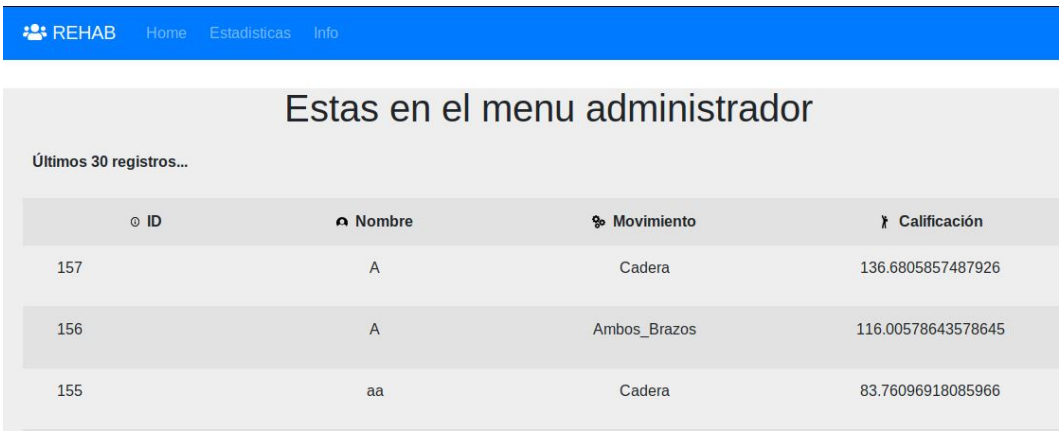

Figura 6.19: Administrador Web

Tras iniciar sesión, el sistema transitará al menú principal 6.19. Este menú cuenta con una barra de navegación que te permite transitar entre el menú de inicio, el menú de monitorización, y adicionalmente, el menú de información del sistema. Inicialmente se visualiza el menú de inicio. En este menú se encuentran los datos de los últimos 30 movimientos que se han hecho en el sistema, independientemente de que paciente lo haya hecho.

Figura 6.20: Administrador Web

Si se hace click en la barra de navegación en la pestaña de estadísticas, visualizarás una interfaz como la mostrada en 6.20. Como se puede observar justo debajo del encabezado, hay dos selectores desplegables y un botón. El primer selector te proporciona una lista con los diferentes usuarios que hay en el sistema. En el que debes elegir el paciente del que quieres ver la evolución.

El segundo selector contiene los diferentes movimientos que el paciente debe hacer, y adicionalmente otra opción denominada Todos. Está última opción sirve para obtener una visión general de como evoluciona el paciente independientemente del movimiento que realice. Por tanto, primero se selecciona el usuario y después el movimiento a monitorizar. Después se pulsa el botón Acceder a Estadísticas del Usuario y el sistema proporciona gráficamente la evolución del paciente.

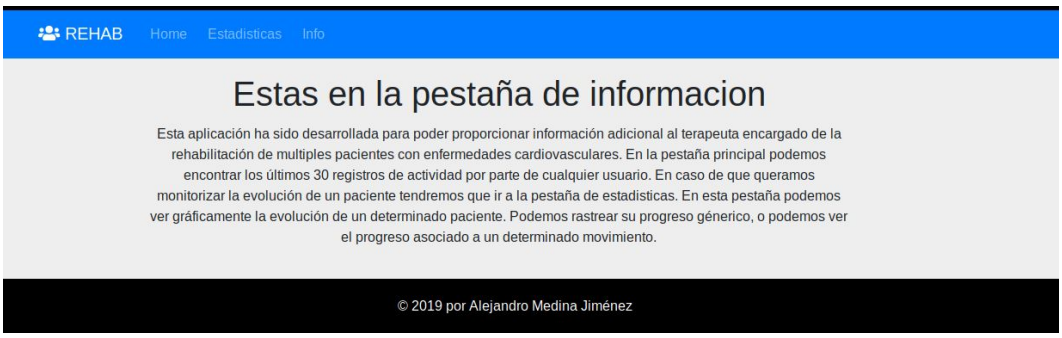

Figura 6.21: Administrador Web

6.2. DISTRIBUCIÓN DEL TRABAJO

A continuación se describen las iteraciones en las cuales se ha desarrollado el prototipo funcional del proyecto. El desarrollo ha sido totalmente secuencial ya que el equipo de desarrollo está compuesto por una única persona. Esto implica que ninguna iteración se ha solapado con otra.

1 # Familiarización con las tecnologías	
Historia de Usuario	-
Tiempo Asignado	60 horas
Durante este proyecto se han empleado diferentes lenguajes de programación y dispositivos hardware diferentes. Esto ha implicado una curva de aprendizaje bastante pronunciada. Para el desarrollo de este proyecto a sido necesario utilizar Unity, y por tanto aprender el lenguaje de programación C#. Por otro lado tambien ha sido necesario aprender desarrollo en Android, todo necesario para la implementación del software en los dispositivos Android. Por otra parte, ha sido necesario aprender el uso de Zero C Ice, python, R y Sql para la implementación del servidor que se comunica con los clientes Android. Y finalmente, para el desarrollo de la parte web ha sido necesario HTML, CSS, Javascript puro y un framework denominado KnockoutJS y Flask para el servidor web.	

Figura 6.22: Iteración - 1

2 # Definir la arquitectura del servidor en ZeroC Ice	
Historia de Usuario	9
Tiempo Asignado	20 horas
Para permitir que el sistema sea facilmente escalable, se ha definido una estructura en ZeroC Ice compuesta por diferentes agentes. En está iteración se desarrolla el rol que va a jugar cada agente y la forma que van a tener de interactuar entre ellos. Se definen los métodos que utilizarán para realizar las comunicaciones y se balancea la carga entre los agentes. Tambien se crean los ficheros de configuración de la plataforma de Ice y la estructura de directorios del servidor.	

Figura 6.23: Iteración - 2

3 # Crear la aplicación utilizada por el paciente	
Historia de Usuario	2
Tiempo Asignado	80 horas

Para crear la versión inicial de la aplicación utilizada por el paciente, se ha utilizado con base un proyecto vacio en Unity.

A este proyecto se le ha incorporado el paquete de Nuitrack para gestionar los gestos y el esqueleto del paciente. Tambien se ha añadido un cliente TCP que realiza la comunicación con un prototipo de agente en el servidor. El cliente TCP envia los datos que recopila del usuario a través del paquete Nuitrack, pero una vez en el servidor los datos aun no son utilizados.

Por otro lado, se diseñan todas las escenas que intervienen en la aplicación, asi como las interfaces y botones necesarios para que el cliente interactue con la aplicación.

Figura 6.24: Iteración - 3

4 # Incorporar movimientos al sistema	
Historia de Usuario	7
Tiempo Asignado	120 horas

En esta iteración se define de cara al servidor el agente que gestiona la conexión TCP y el agente gestor de la base de datos. Tambien se definen las tablas que componen la base de datos y el tipo de datos que habrá en ellas.

Con el objetivo de poder incorporar nuevos movimientos, se define la aplicación Android que utilizará el terapeuta. En está iteración lo único que permite realizar está aplicación es desde un menú de inicio, incorporar un nuevo movimiento al sistema, sin especificar el usuario que lo incorporó. Para ello es necesario al igual que se ha hecho en el cliente que utilizará el paciente, crear un cliente TCP que envie datos al servidor y hacer uso del paquete Nuitrack para recopilar los datos que serán enviados.

Figura 6.25: Iteración - 4

5 # Incorporar nuevos usuarios en el sistema	
Historia de Usuario	4
Tiempo Asignado	20 horas

Ahora que es posible incorporar nuevos movimientos al sistema, es el momento de permitir la creación de usuarios para que realicen los movimientos que se han registrado. Para ello, se definen las tablas relacionadas con los usuarios. Despues se incorporan al agente gestor de bases de datos el código necesario para que el mismo pueda manipular las tablas creadas previamente, realizando consultas para obtener los usuarios, añadir nuevos o eliminarlos. Tambien es necesario que el agente en el que se delega la función de la gestión con el cliente TCP sea capaz de interpretar cuando llega un usuario nuevo al sistema para su correspondiente registro.

Figura 6.26: Iteración - 5

6 # Clasificar los movimientos realizados por los pacientes	
Historia de Usuario	3
Tiempo Asignado	40 horas

Clasificar los movimientos de los pacientes es una tarea que requiere mucho análisis. Es necesario realizar un largo proceso de pruebas para garantizar que el sistema funciona correctamente.

Para concluir está iteración se han implementado los agentes supervisor y clasificador. El agente supervisor no sería necesario ya que no se va a realizar el análisis para obtener una nota aún. Sin embargo, se decidió incorporarlo en este punto debido a que de este modo la carga de estos dos agentes queda balanceada. El agente clasificador es el encargado de determinar que serie del movimiento que ha realizado el paciente, hay que comparar con una de las extraídas de la base de datos utilizadas a modo de referencia. Tras determinar cuales son comparadas, estás dos series traducidas en formato de vector son enviadas al agente supervisor, que responde con una nota. En base a las notas que va recuperando el agente clasificador del agente supervisor se va construyendo un vector de distancias y en base a unas métricas se determina que movimiento es el que ha realizado en función de los movimientos con los que ha sido comparado.

Figura 6.27: Iteración - 6

7 # Almacenar la nota del paciente para monitorizarlo posteriormente	
Historia de Usuario	5
Tiempo Asignado	40 horas

Hasta este momento solo se ha implementado la clasificación del movimiento para determinar el movimiento realizado por el paciente. En este momento se implementa toda la lógica que determinará la nota que el paciente ha obtenido. Esta nota se almacena posteriormente para luego ser analizada por el terapeuta.

Para ello ha sido necesario implementar el resto de metodos del agente supervisor, del agente gestor de base de datos y coordinarlos con el agente que gestiona la conexión con el cliente. El agente supervisor ha implementado el método en el que filtra las articulaciones que no son relevantes a la hora de analizar la nota del movimiento. El método en el que realiza el calculo de la nota y el método en el que se comunica la calificación, junto al movimiento realizado y el nombre del usuario al agente gestor de bases de datos. Por otro lado, los métodos del agente gestor en los que incorpora los registros de la actividad del paciente a la base de datos. Finalmente, tambien es necesaria implementar la lógica por la que el agente gestor de la conexión comunica los datos solicitados al agente supervisor. La parte en la que se le comunica la nota al usuario fue implementada en la iteración anterior.

Figura 6.28: Iteración - 7

8 # Monitorizar evolución del paciente	
Historia de Usuario	6
Tiempo Asignado	60 horas

Llegando a este punto, el sistema permite incorporar usuarios, movimientos, clasificar movimientos y analizarlos. Pero aún no es posible realizar un seguimiento de los movimientos realizados por un determinado usuario.

En está iteración se implementa toda la gestión sobre la evolución de los pacientes. Para ello se definen en la base de datos nuevos campos con los credenciales de acceso del terapeuta. Se implementan todas las operaciones que el cliente web podría utilizar en el servidor y se testean utilizando Postman. Tras verificar que los datos se recuperan bien del back-end, se define el cliente web. Se crean varías vistas, el login, el panel de inicio, el panel de configuración y el panel del que se recuperan las estadísticas. Se utiliza la librería de bootstrap con el objetivo de facilitar el diseño, y la librería Chart.js para visualizar los datos recopilados por los usuarios en forma de gráficos.

Figura 6.29: Iteración - 8

9 # Obtener resultados tras realizar un movimiento	
Historia de Usuario	8
Tiempo Asignado	20 horas

El cliente puede obtener una calificación tras realizar un ejercicio, y el paciente puede revisar la evolución del paciente. El problema es que el cliente no tiene porque ser capaz de interpretar la nota proporcionada por el sistema, y para poder realizar correctamente los ejercicios de rehabilitación sería conveniente que el sistema le proporcione información que indique el estado de su progreso.

Por tanto es en este punto en el que se implementan los agentes motivacionales cuyo objetivo es traducir toda está información para que el usuario final sea capaz de comprenderla. Tras obtener la nota y en base a los movimientos realizados previamente, el sistema realiza una estimación de los movimientos que serían convenientes que el paciente realizará. Además tambien envía al paciente una traducción en lenguaje coloquial sobre lo bien que ha realizado el ejercicio.

Figura 6.30: Iteración - 9

10 # Pruebas de integración	
Historia de Usuario	1
Tiempo Asignado	60 horas

Finalmente, con todos los requisitos del sistema ya implementados, se procede a realizar las pruebas de integración.

En estás pruebas se comprueba que todo el sistema funcione de una forma correcta y coordinada. Se recupera información de debug de los diferentes agentes y clientes que componen el sistema para verificar que el paso de datos entre ellos se ejecuta de la manera más óptima. Tras realizar varias pruebas y verificar que el sistema cumple con los requisitos, se procese a hacer una limpieza en el código para preparar el prototipo final.

Figura 6.31: Iteración - 10

6.3. COSTO DE DESARROLLO

Este proyecto ha sido desarrollado entre el 20 de mayo de 2019 y el 20 de noviembre de 2019, por lo que han sido 6 meses de trabajo. En la figura 6.32 se pueden observar los cálculos necesarios para realizar el coste. A continuación se desarrollaran los gastos que ha implicado el desarrollo de este proyecto.

Presupuesto de Proyecto

Proyecto	Sistema Multiagente para Rehabilitación Física de Pacientes con Enfermedades Neurológicas.	Duracion del proyecto	6 meses
Lider	David Vallejo Fernández y Alejandro Medina Jiménez		

Costos directos	$	21.690,45
Costos indirectos	$	15,00
Reserva para riesgos		15%

Presupuesto	$	21.705,45
Riesgo	$	3.255,82
Total	$	24.961,27

Costos Directos

Elemento	Tipo de recurso	Tipo de Unidad	Unidades	Precio por unidad	Costo
Salario de un desarrollador	Gastos de personal	hora	672	30 €	20160
Licencia Software Nuitrack Pro	Materiales	euros	120	1	120
Hosting + Dominio	Materiales	euros	7,95	1	7,95
Orbec Persee Tvico	Equipamiento	euros	252,5	1	252,5
Portatil Asus ROG	Equipamiento	euros	1150	1	1150

Costos Indirectos

Elemento	Tipo de recurso	Tipo de Unidad	Unidades	Precio por unidad	Costo Indirecto
Paquete Reconocimiento Voz Unity	Materiales	euros	15	1	15

Figura 6.32: Costos del desarrollo del proyecto.

Para calcular el coste del proyecto hay que tener en cuenta los costes directos e indirectos.

- **Costes directos:**
  - Salario de empleados: el salario de un desarrollador que ha trabajado durante 6 meses de lunes a viernes, 4 horas al día. Esto hace un total de 672 horas de desarrollo. El coste de la hora del desarrollador son 30€, lo que hace un total de 20.160€.
  - Licencias software: han sido necesarios varios componentes software para realizar este proyecto. Una licencia Nuitrack Pro utilizada por los dispositivos de los clientes que utilizan los pacientes y el administrador. Esto supone un coste de 120€ en un único pago. Por otro lado el precio del dominio del servidor web con un coste de 7.95€. El alojamiento va implícito ya que es instalado en un equipo portátil.
  - Equipamiento: El dispositivo Android utilizado por el cliente y el administrador del sistema con un coste de 252,50€. Un portátil en el que se ejecuta el servidor basado en Ice y el servidor web con un coste de 1150€.
- **Costes indirectos:**
  - Paquetes software no incluido en el proyecto: para satisfacer una historia de usuario que luego fue modificada, se compró un paquete para Unity. Este paquete permitía realizar reconocimiento por voz, pero finalmente no fue compatible con el hardware del dispositivo.
  - Horas extra del desarrollador: han sido necesarias 2 semanas adicionales de trabajo para realizar más pruebas al proyecto. Esto supone un gasto de 40 horas del desarrollador. Esto tiene un coste de 1200€.

CAPÍTULO 7

CONCLUSIONES

En este capitulo se hará una discusión sobre los resultados obtenidos. En primer lugar se indicará el grado de satisfacción en el que cada objetivo específico ha sido cumplido. Después se propondrán una serie de mejoras para una futura línea de trabajo. Finalmente el autor de este proyecto expondrá su opinión personal.

7.1. OBJETIVOS ALCANZADOS

En esta sección se hablará de los objetivos planteados en el capítulo 2. Para ello, se diferencian entre objetivos generales y específicos.

7.1.1. Objetivos generales

- El paciente, cuando active la herramienta verá los ejercicios que tiene que realizar para ir completando el tratamiento, a medida que los vaya ejecutando, el sistema le proporcionará información acerca de la exactitud de sus movimientos con respecto a lo estipulado por el clínico.

  ¿Alcanzado? SI. Se ha llevado un estudio acerca de como realizar la comparación de los movimientos, y se ha logrado proporcionar los resultados al paciente. Por otro lado, el paciente ha de saber por el clínico los movimientos que debe realizar.

- El clínico, podrá almacenar los diferentes movimientos que quiere que ejecute el paciente, en función del tratamiento que necesite determinará los diferentes movimientos a los distintos pacientes, además de una vez que el paciente ha finalizado sus ejercicios poder ver con que exactitud los ha realizado, además el sistema le recomendará nuevos movimientos asociados a la enfermedad específica del paciente.

  ¿Alcanzado? SI. Se ha desarrollado un sistema que permite la gestión de los diferentes movimientos que el paciente podría realizar en función del tratamiento acordado. Cuando el paciente finaliza un ejercicio, el sistema le proporciona una nota y además le recomienda realizar nuevos movimientos al paciente, por tanto se ha alcanzado el objetivo.

7.1.2. Objetivos específicos

- Reducir la interacción con los pacientes gracias al clasificador automático, evitando la interacción explícita, que suele dar lugar a confusión y errores.

  ¿Alcanzado? SI. El sistema proporciona al paciente una interfaz muy sencilla de utilizar con muy poca interacción. No tiene que especificar el movimiento que va a realizar, lo realiza y el sistema calcula que movimiento ha realizado. Esto funciona con un grado de satisfacción

bastante bueno siempre que las condiciones lumínicas sean adecuadas y la distancia con el dispositivo sea correcta. En el momento que falla una de las dos condiciones la calidad del análisis se ve afectada exponencialmente.

- Facilidad de uso, utilizando un diseño simple que un usuario inexperto pueda aprender sin problemas. Uno de los propósitos que se pretendía satisfacer era el diseño de una interfaz sencilla e intuitiva, para que cualquier persona pueda utilizarla sin problemas. Esto se ha conseguido permitiendo a los usuarios ver las imágenes generadas por la cámara, permitiendo también que la mayoría de las opciones se activen por comandos de voz para que los pacientes con movilidad reducida no tuviesen que depender del teclado y del ratón u otros periféricos.

  ¿Alcanzado? SI, PARCIALMENTE. El sistema está dotado de una sencilla interfaz tanto para el paciente como para el administrador del sistema. Por un lado, la interfaz del paciente solo tiene tres posibles formas de interacción, cerrar la mano derecha, la izquierda o ambas. Con estos sencillos gestos es posible navegar por la interfaz sin necesidad de utilizar otro tipo de interacción. También permite al usuario observar como realiza el movimiento de la cámara y además observar como la cámara reconoce su esqueleto. Este objetivo es cumplido parcialmente debido a que en un principio, la interacción debía ser por voz. Este objetivo no ha sido posible cumplirlo por limitaciones propias del hardware. En su defecto, se ha incorporado al sistema la interacción gestual, que es una alternativa muy eficaz al reconocimiento por voz. Por otra parte, la interfaz web posee un sistema de inicio de sesión con dos únicos campos. Una vez dentro de la aplicación, tiene un sencillo panel de inicio con una visión de los últimos movimientos registrados, un panel de ayuda y una interfaz para monitorizar la rehabilitación de los pacientes. En esta interfaz hay dos selectores, uno para elegir al usuario y otro para elegir el movimiento. Con estos datos, el sistema muestra estadísticas de uso de la aplicación del paciente y el movimiento seleccionado.

- Motivación a seguir utilizando la herramienta, para evitar que el paciente deje de utilizarla durante el tratamiento. Es importante la actitud del paciente con la herramienta ya que éste no puede interrumpir el tratamiento, ya el proceso de rehabilitación debe de ser continuo hasta su finalización. Para motivar a los pacientes se llevará a cabo un seguimiento realizado por el clínico donde se podrá ver el estado inicial, los ejercicios a realizar, los últimos avances, el estado de la zona afectada. El paciente al ver una evolución en su recuperación se mantendrá constante con el tratamiento.

  ¿Alcanzado? SI, PARCIALMENTE. Para contribuir a la motivación durante la realización de los ejercicios de rehabilitación del paciente, se pretendía incorporar un reconocimiento facial que obtendría datos sobre el estado de ánimo del paciente. Esto no ha sido posible debido a que la librería utilizada durante el desarrollo, aún no ha sacado la versión para el dispositivo con el que se ha realizado este proyecto. Por tanto, se hablará de ello en trabajos futuros, como una posible mejora. Por otra parte, con el objetivo de proporcionar información que motive al paciente, se le proporciona una nota y un mensaje indicando como de bien se han realizado los ejercicios.

- Escalabilidad, permitiendo así añadir más agentes por si en algún futuro fuese necesario añadir más funcionalidad o ampliar el sistema.

  ¿Alcanzado? SI. El sistema permite ampliar la funcionalidad con un trabajo mínimo durante la integración del nuevo agente. Esto se ha logrado gracias al uso de un framework como ZeroC Ice, para el diseño de un sistema multi-agente.

7.2. PROBLEMAS DURANTE EL DESARROLLO

El hecho de implementar un sistema en el que intervienen varios dispositivos intercambiando datos entre ellos supone un reto. El desarrollo del proyecto ha causado una serie de problemas que han se han ido resolviendo secuencialmente.

- **Complejidad de implementar un sistema heterogéneo:** con diferencia el mayor problema ha sido abordar una solución en la que intervienen dispositivos hardware diferentes que implican utilizar diferentes tecnologías. Ya que cada uno tiene una curva de aprendizaje propia de la forma en la que se desarrolla el software. Esto a su vez ha requerido del uso de diferentes lenguajes de programación y posteriormente realizar una correcta integración del sistema en conjunto.

- **Conexión entre el cliente Android y el servidor:** en una primera etapa del proyecto, se intentaron utilizar los clientes Android como otro agente dentro de la plataforma de ZeroC Ice. Tras investigar sobre el problema se decidió que a nivel de tiempo no era viable debido a la tediosa curva de aprendizaje necesaria para llevar a cabo está conexión. Por tanto, para resolver el problema se implementó un agente encargado de realizar está conexión por TCP con los clientes. El resultado es similar, pero a nivel de tiempo permitió reducir el tiempo del proyecto de manera considerable.

- **Integración del agente clasificador con R:** la librería que permite realizar los cálculos necesarios durante la clasificación solo está disponible en R. Por tanto, fue necesario aprender a desplegar un servidor R que atienda peticiones. Una vez hecho esto, realizar la conexión entre un programa basado en python y este servidor obteniendo la funcionalidad deseada.

- **Integración del reconocimiento por voz:** Este ha sido uno de los puntos del proyecto donde se ha perdido con diferencia más tiempo. Esto es debido a que el dispositivo está dotado de un micrófono que funciona correctamente. Para llevar a cabo en un primer momento está tarea, se utilizó la API de google creando una aplicación demo, probada en un móvil previamente con buenos resultados, y después siendo instalada en el dispositivo para su posterior integración en el proyecto. Los resultados obtenidos daban lugar a confusión, ya que aparentemente estaba funcionando correctamente, pero no daba ningún resultado. Después se intentó llevar a cabo este objetivo a través de la API de reconocimiento por voz de Microsoft Azure. Los resultados fueron similares a los de la prueba anterior. Finalmente, se compró un plugin en la tienda de Unity, que daba está funcionalidad ya implementada. Se creó una demo al igual que en los otros casos y se instaló en un móvil y el dispositivo del cliente. En el móvil funcionaba sin problemas, pero en el dispositivo se obtuvo mensaje indicando que no era compatible. Tras investigar, se llegó a la conclusión de que era incompatible con este dispositivo. Está afirmación fue extraída de la comunidad de Nuitrack, de un desarrollador que pasó por una experiencia similar.

- **Integración del reconocimiento facial:** Debido a uno de los objetivos del proyecto se pretendió implementar un reconocimiento de la expresión facial del usuario. Para ello se diseñó una demo que funcionaba sin proporcionar ningún error, salvo que no reconocía la cara. Tras investigar por los foros que era un problema que habían tenido varias personas, un moderador del foro especificó en un mensaje que está función solo estaba disponible en determinados dispositivos. Desafortunadamente el utilizado en este proyecto no permitía realizar esa función, aun que era candidato a implementarla en futuras actualizaciones.

- **Integración con el reconocimiento gestual:** Implementar la interacción gestual ha llevado un proceso de aprendizaje, en el cual ha sido necesario realizar previamente una demo en la que interpretar como funciona este paquete de Nuitrack. Tras varios días de pruebas, se logró capturar el evento que permite reconocer si las manos del usuario están abiertas o cerradas.

7.3. COMPETENCIAS DE LA INTENSIFICACIÓN ALCANZADAS

A continuación se expondrán una serie de competencias que han sido alcanzadas con la realización de este proyecto.

- Capacidad para evaluar la complejidad computacional de un problema, conocer estrategias algorítmicas que puedan conducir a su resolución y recomendar, desarrollar e implementar aquella que garantice el mejor rendimiento de acuerdo con los requisitos establecidos.

  Durante este proyecto, ha sido necesario realizar un trabajo de investigación sobre como evaluar series temporales. Está investigación ha requerido de un estudio de la complejidad de el problema de analizar dos series temporales en tiempo real. Se han evaluado diferentes alternativas sobre algoritmos que trabajan con series temporales y finalmente se decidió utilizar el algoritmo DTW. Por tanto está competencia ha sido cumplida en este proyecto.

- Capacidad para conocer los fundamentos, paradigmas y técnicas propias de los sistemas inteligentes y analizar, diseñar y construir sistemas, servicios y aplicaciones informáticas que utilicen dichas técnicas en cualquier ámbito de aplicación.

  El hecho de que el propio título del proyecto incluya la palabra sistema multi-agente demuestra que está competencia se cumple. Esto se debe a que el servidor es un sistema multi-agente compuesto por una comunidad de agentes que cooperan para realizar un seguimiento al paciente que utiliza la aplicación.

- Capacidad para adquirir, obtener, formalizar y representar el conocimiento humano en una forma computable para la resolución de problemas mediante un sistema informático en cualquier ámbito de aplicación, particularmente los relacionados con aspectos de computación, percepción y actuación en ambientes entornos inteligentes.

  La cámara que posee el hardware de la aplicación cliente transforma la representación del cuerpo humano que reconoce cuando alguien se pone delante de ella, en coordenadas tridimensionales. Posteriormente estás coordenadas se envían al servidor, se procesan y se van almacenando en el sistema en forma de vectores para ser comparadas con otros movimientos y deducir de que movimiento se trata. Por tanto obteniendo las posiciones de una persona y representándolas de un modo computable se puede resolver un problema, que es identificar un movimiento.

- Capacidad para desarrollar y evaluar sistemas interactivos y de presentación de información compleja y su aplicación a la resolución de problemas de diseño de interacción persona computadora.

  El hecho de utilizar una interfaz en la aplicación cliente, adaptada a pacientes en fase de rehabilitación demuestra que se ha sabido aplicar está competencia en un proyecto real. Además este proyecto se ha orientado de modo que la persona que utilice la aplicación cliente tenga la menor interacción explícita posible, dejando que sea el propio sistema el que deduzca la información que no es solicitada al paciente. Por ejemplo, no se le solicita conocer el movimiento que va a realizar.

7.4. POSIBLES TRABAJOS FUTUROS

A continuación se propondrán una serie de posibles líneas en las que orientar el desarrollo del proyecto.

7.4.1. Incorporar reconocimiento facial al sistema

Esto permitiría recopilar una gran cantidad de información sobre los usuarios con multitud de aplicaciones. Entre ellas, podría hacerse un análisis del estado de ánimo del paciente durante sus ejercicios de rehabilitación. Para incorporar reconocimiento facial al sistema habría que seguir una serie de pasos. En un primer momento esto se podría hacer únicamente en la aplicación del cliente, ya que no tiene mucho sentido hacerlo en la aplicación del terapeuta. Por tanto habría que modificar el proyecto del cliente del paciente, administrador web y el servidor para que interprete este nuevo tipo de peticiones.

- Cliente del paciente: Este es el punto más delicado ya que el se encargaría de enviar al servidor los datos que recoja del usuario que hay delante de la cámara. Estructurar correctamente los datos en este punto puede hacer que el desarrollo del servidor sea más sencillo. El problema es que para poder llevar a cabo este trabajo, sería necesario que los desarrolladores de la plataforma de Nuitrack sacarán una nueva versión de la aplicación a la que este cliente realiza las peticiones. La ventaja es que está librería ya está implementada en otros dispositivos de una gama superior. Se podría contemplar la posibilidad de utilizar uno de estos dispositivos para hacer este reconocimiento.
- Administrador web: Del mismo modo que ahora se envían nuevos datos al servidor para que sean interpretados, el administrador web también requiere de una mejora para poder interpretar estos nuevos registros. Una buena opción sería mostrar cada movimiento que hace el paciente como un registro similar a los de la pestaña de inicio, con una nueva columna en la que aparezca un emoticono con una expresión o algo similar.
- Servidor: Un buen primer comienzo para modificar el servidor sería incorporar nuevas tablas a la base de datos donde se almacenen este tipo de registros con una relación uno a uno con los identificadores de cada registro de los movimientos hechos por los pacientes. Después habría que incorporar un nuevo agente al sistema encargado de interpretar las diferentes instancias de los gestos del paciente en el tiempo para obtener conclusiones. Finalmente habría que programar la comunicación entre el agente gestor de bases de datos, la pizarra, el agente de captura del movimiento y el agente de feedback con el nuevo agente.

7.4.2. Hacer los clientes Android compatibles con más dispositivos

En estos últimos años el hardware de los dispositivos móviles ha evolucionado exponencialmente. Estos dispositivos ahora permiten realizar cosas que antes requerían un elevado coste computacional, como por ejemplo realidad aumentada (filtros de Instagram). Actualmente el paquete Nuitrack puede ser instalado en varios dispositivos móviles del mercado. Los propios desarrolladores de Nuitrack han hecho una lista con una serie de dispositivos con los que la aplicación ha sido probada:

- Odroid XU3 (Android 4.4.4 Kitkat) (rooteado)
- Samsung Galaxy S4 (Android 5 Lollipop) (no rooteado)
- Samsung Galaxy S5 (Android 6 Marshmallow) (no rooteado)

- Samsung Galaxy S6 (Android 7 Nougat) (no rooteado)
- Samsung Galaxy S8 / S8 + (Android 8 Oreo) (no rooteado)

La mayoría de móviles que aparecen en la lista, tienen un hardware que en su año fue considerado gama alta. Actualmente un dispositivo con estas prestaciones puede ser adquirido por un coste realmente bajo respecto a una gama media. Por lo que una mejora importante en el proyecto podría ser probarla con todo tipo de dispositivos móviles para que los pacientes puedan realizar sus ejercicios de rehabilitación simplemente con su móvil y un trípode. Con esta mejora aumentaría la compatibilidad con diferentes dispositivos y esto permitiría que la aplicación llegue a un mayor número de usuarios.

7.4.3. Incorporar reconocimiento por voz

Asumiendo que el punto anterior se llevará a cabo, esto permitiría incorporar reconocimiento por voz en las aplicaciones. Respecto a esta mejora, destaca lo sencilla que podría hacerse la comunicación con el usuario. Además se podría estudiar el uso del reconocimiento de la voz del usuario a modo de contraseña para que el paciente acceda a realizar el movimiento.

Incluso podría hacerse todo esto de forma implícita. Para ello sería necesario que el paciente le dijera alguna clave al sistema, y el propio sistema podría clasificar la voz del usuario utilizando el mismo algoritmo que el que se utiliza para comparar un movimiento. De este modo el sistema reconocería al usuario con solo escuchar su voz y esto podría ser un modo de inicio de sesión bastante seguro e intuitivo de usar.

Por otra parte, permitiría al sistema ampliar el abanico de movimientos disponibles. Esto se debe a que el hecho de tener que utilizar las manos para transitar por el sistema limita parcialmente el número de movimientos que el sistema podría tener. Por ejemplo un movimiento de abrir y cerrar el puño actualmente no sería posible, en cambio si se implementará reconocimiento por voz, se podrían analizar los gestos de la mano del usuario para obtener conclusiones a nivel de rehabilitación. Actualmente los gestos son usados para transitar por el sistema.

7.4.4. Incorporar nuevos movimientos

En el prototipo expuesto en este proyecto, hay cinco movimientos diferentes en total. En un futuro sería conveniente incorporar nuevos movimientos que el sistema pueda analizar y comparar. Asumiendo que el reconocimiento por voz se llevará acabo, la tecnología que se utiliza durante el proyecto da mucho potencial de escalabilidad.

Ya que el reconocimiento de la posición de las manos, conocer si las manos están abiertas o cerradas, y la posición tridimensional de cada articulación del cuerpo junto con la rotación de cada articulación, proporcionan una gran cantidad de información. Está información sería suficiente para que se pudieran diferenciar con éxito un número más elevado de movimientos respecto a los disponibles en el sistema actual.

Además, el prototipo del proyecto no contempla la posición de las piernas durante el reconocimiento del cuerpo, ya que no se han registrado movimientos en los que su intervención sea fundamental. Está sería una línea de trabajo sencilla de implementar y en la que no sería muy complicado conseguir que el sistema diferencie los movimientos con éxito.

7.4.5. Incorporar técnicas de aprendizaje automático

Otra posible línea de desarrollo sería incorporar en el sistema técnicas de aprendizaje automático que permitan al sistema aprender de sus errores. Actualmente si el sistema se equivoca no tiene

ningún tipo de repercusión. Una posible solución sería incorporar en el cliente Android administrador, en la opción de comparar movimiento, otra interfaz. En esta interfaz, tras obtener el resultado del movimiento que ha reconocido el sistema, se podría indicar si el movimiento ha sido realizado correctamente o no. De este modo, se podrían recopilar los datos necesarios para entrenar una red neuronal por aprendizaje supervisado y mejorar la experiencia de usuario.

7.5. OPINIÓN PERSONAL DEL AUTOR

El impacto que podría tener la implantación del sistema propuesto en centros sanitarios no puede medirse hasta que el proyecto pase a un entorno en producción. Este trabajo ha sido útil para entender como trabajan las series temporales y sus posibles aplicaciones, para crear un sistema heterogéneo y ver como integrar los diferentes elementos que lo componen correctamente.

Me ha resultado muy enriquecedor trabajar con Unity, me ha parecido una herramienta muy potente que abre literalmente un mundo de posibilidades. También hay que decir que tiene una curva de aprendizaje alta y me ha costado bastante entenderlo. Sobre todo la parte asociada al desarrollo utilizando Unity y Android, y probando la aplicación que utilizará el paciente ha sido gratificante ver que realmente es capaz de clasificar y evaluar de una manera adecuada un movimiento.

Desde un punto de vista práctico, la elección de ZeroC Ice para hacer la plataforma multi-agente me ha parecido todo un acierto. Esto se debe a la gran flexibilidad de este marco durante el desarrollo, me habría gustado integrar completamente los clientes Android en el sistema ZeroC Ice, y no tener que conectarlos con un nodo del sistema multi-agente a través de sockets. El problema es que de haberlo hecho como quería, los tiempos de desarrollo habrían aumentado bastante si se hubiera integrado finalmente. Creo que el sistema funciona bastante bien, pero que se podría mejorar más aún utilizando otro tipo de técnicas a modo de complemento del algoritmo DTW. Por otra parte, también hay que destacar que utilizar Nuitrack para hacer el reconocimiento del paciente es todo un acierto en comparación con otras soluciones como Kinect.

Estoy convencido de que podría perfeccionar más aún este sistema, ya que el tiempo de desarrollo de este proyecto se ha prolongado excesivamente. Durante el tiempo que lo he estado implementando he aprendido muchísimo y probablemente si empezará a hacerlo ahora, tardaría mucho menos que antes y adoptaría una solución mejor. Pero hay que tener en cuenta la cantidad de recursos invertida, para medir con objetividad la calidad obtenida.

BIBLIOGRAFÍA

[1] Lluís Tomás Abadal. «La evolución de los programas de ejercicio físico en el ámbito de la salud». En: *Prescripción del ejercicio físico para la salud. Editorial Paidotribo* (1996).

[2] Fabio Luigi Bellifemine, Giovanni Caire y Dominic Greenwood. *Developing multi-agent systems with JADE*. Vol. 7. John Wiley & Sons, 2007.

[3] Fabio Bellifemine, Agostino Poggi y Giovanni Rimassa. «Developing multi-agent systems with JADE». En: *International Workshop on Agent Theories, Architectures, and Languages*. Springer. 2000, págs. 89-103.

[4] Miguel Alfredo Bustos, Norma Beatriz Perez y Mario Berón. «Plataformas para el desarrollo de aplicaciones móviles». En: *XVII Workshop de Investigadores en Ciencias de la Computación (Salta, 2015)*. 2015.

[5] José H Canós y Mª Carmen Penadés Patricio Letelier. «Metodologías ágiles en el desarrollo de software». En: (2012).

[6] B. Cascales y P. Lucas. *El Libro de LaTeX*. Pearson Education, 2005. ISBN: 9788420537795.

[7] B. Cascales y col. *LaTeX. Una imprenta en sus manos*. Aula Documental de Investigación, 2000.

[8] «Evaluación del ejercicio de rehabilitación utilizando sensores inerciales: un estudio analítico transversal». En: *Revista de neuroingeniería y rehabilitación* (). Ed. por BioMed Central.

[9] Jonathan D Cryer y Kung-Sik Chan. *Time series analysis. With applications in R*. Springer, 2008.

[10] Dorothy F Edwards y col. «Screening patients with stroke for rehabilitation needs: validation of the post-stroke rehabilitation guidelines». En: *Neurorehabilitation and Neural Repair* 20.1 (2006), págs. 42-48.

[11] Rick Evertsz y col. «Implementing industrial multi-agent systems using JACK TM». En: *International Workshop on Programming Multi-Agent Systems*. Springer. 2003, págs. 18-48.

[12] Carlos Ángel Iglesias Fernández y Carlos Ángel Iglesias Fernández. «Definición de una metodología para el desarrollo de sistemas multiagente». En: *Enfoques* 35 (1998), págs. 57-190.

[13] Thomas Finnegan. *Learning Unity Android Game Development*. Packt Publishing Ltd, 2015.

[14] Lena Fleig y col. «Exercise maintenance after rehabilitation: How experience can make a difference». En: *Psychology of Sport and Exercise* 12.3 (2011), págs. 293-299.

[15] Liping Fu y col. «Experimental realization of discrete fourier transformation on NMR quantum computers». En: *Applied Magnetic Resonance* 19.2 (2000), págs. 153-159.

[16] MD García-Manzanares y col. «Factores psicosociales y calidad de vida en la rehabilitación médica». En: *Rev Mex Med Fis Rehab* 18.1 (2006), págs. 11-17.

[17] A Gil-Agudo y col. «Experiencia clínica de la aplicación del sistema de realidad TOyRA en la neuro-rehabilitación de pacientes con lesión medular». En: *Rehabilitación* 46.1 (2012), págs. 41-48.

[18] M. Goossens, F. Mittelbach y A. Samarin. *The LaTeX companion*. 2.ª ed. Addison-Wesley Reading, MA, 2004.

[19] M. Goossens, S. Rahtz y F. Mittelbach. *The LaTeX graphics companion*. 2.ª ed. Addison-Wesley Reading, MA, 2007.

[20] G.A. Grätzer. *First steps in LaTeX*. 1.ª ed. Springer Verlag, 1999.

[21] G.A. Grätzer. *More math into LaTeX*. 4.ª ed. Birkhauser, 2007.

[22] Dipalee Gupta y Siddhartha Choubey. «Discrete wavelet transform for image processing». En: *International Journal of Emerging Technology and Advanced Engineering* 4.3 (2015), págs. 598-602.

[23] Olivier Gutknecht y Jacques Ferber. «MadKit: Organizing heterogeneity with groups in a platform for multiple multi-agent systems». En: *Rapport Interne LIRMM* 97188 (1997), pág. 1997.

[24] Pedro Hípola, Benjamín Vargas-Quesada y col. «Agentes inteligentes: definición y tipologia. Los agentes de información». En: (1999).

[25] MC Jiménez y col. «Telemedicina aplicada al ictus en las Islas Baleares: el proyecto Teleictus balear». En: *Revista de Neurología* 54.1 (2012), págs. 31-40.

[26] José Joskowicz. «Reglas y prácticas en eXtreme Programming». En: *Universidad de Vigo* 22 (2008).

[27] H. Kopka y P.W. Daly. *A guide to LaTeX*. 4.ª ed. Addison-Wesley, 2004.

[28] Pedro Salcedo Lagos. «Inteligencia Artificial Distribuida y Razonamiento Basado en Casos en la Arquitectura de un Sistema Basado en el Conocimiento para la Educación a Distancia (SBC-ED).» En: *Chile: Departamento de Metodología de la Investigación e Informática Educativa, Facultad de Educación, Univ. de Concepción* (2003).

[29] L. Lamport. *LaTeX: A document preparation system*. 2.ª ed. Addison-Wesley, 1994.

[30] Danny B Lange y Oshima Mitsuru. *Programming and Deploying Java Mobile Agents Aglets*. Addison-Wesley Longman Publishing Co., Inc., 1998.

[31] Craig Larman. *UML y patrones*. Pearson, 1999.

[32] Patricio Letelier y Mª Carmen Penadés. «Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP)». En: (2006).

[33] R Lloréns y col. «BioTrak: análisis de efectividad y satisfacción de un sistema de realidad virtual para la rehabilitación del equilibrio en pacientes con daño cerebral». En: *Neurología* 28.5 (2013), págs. 268-275.

[34] B Loreto Vergara. «Desarrollo de la Medicina Física y Rehabilitación como especialidad médica». En: (2014).

[35] Matilde Mas y Javier Quesada. *Las nuevas tecnologías y el crecimiento económico en España*. Fundacion BBVA, 2005.

[36] John C Mason y David C Handscomb. *Chebyshev polynomials*. Chapman y Hall/CRC, 2002.

[37] José Alberto Mauricio. «Análisis de series temporales». En: *Universidad Complutence de Madrid* (2007).

[38] Néstor Darío Duque Méndez y Alfonso Pio Agudelo Salazar. «Sistema Multiagente para la Evaluación Personalizada en Cursos Virtuales». En: *Revista Avances en Sistemas e Informática* 3.2 (2006), págs. 7-12.

[39] Daniel Moreno-Zambrano y col. «Enfermedad Cerebrovascular en el Ecuador: Análisis de los últimos 25 años de mortalidad, realidad actual y recomendaciones». En: *Rev Ecuat Neurol* 25.1-3 (2016), págs. 17-20.

[40] Antonio Moreno. «Medical applications of multi-agent systems». En: *Computer Science and Mathematics Department, University of Rovira, Spain* (2003).

[41] Álvaro Moyano. «El accidente cerebrovascular desde la mirada del rehabilitador». En: *Rev Hosp Clín Univ Chile* 21 (2010), págs. 348-55.

[42] Ignacio Bravo Muñoz y col. «Solución inalámbrica para la implementación de un sistema de telemedicina». En: *Novática: Revista de la Asociación de Técnicos de Informática* 177 (2005), págs. 31-33.

[43] Hyacinth S Nwana, Divine T Ndumu, Lyndon C Lee y col. «ZEUS: An advanced tool-kit for engineering distributed multi-agent systems». En: *Proceedings of PAAM*. Vol. 98. 1998, págs. 377-391.

[44] T. Oetiker y col. *The not so short introduction to LaTeX2e*. 2006.

[45] «Organización Mundial de la Salud». En: ().

[46] B Orgun y J Vu. «HL7 ontology and mobile agents for interoperability in heterogeneous medical information systems». En: *Computers in biology and medicine* 36.7-8 (2006), págs. 817-836.

[47] Jose M Pérez y col. «Multi-Agent System (GerMAS) Used to Identify Medicines in Geriatric Residences». En: *Highlights in Practical Applications of Agents and Multiagent Systems*. Springer, 2011, págs. 299-306.

[48] Pincay Piguave y Genesis Jesus. «Proceso de atencion de enfermeria en rehabilitacion de paciente con accidente cerebro vascular.» En: (2018).

[49] David Robledo Fernández. *Desarrollo de aplicaciones para Android II*. Ministerio de Educación, 2014.

[50] Ira Rudowsky. «Intelligent agents». En: *The Communications of the Association for Information Systems* 14.1 (2004), pág. 48.

[51] Jesús Salido. *Curso de LaTeX2e*. Universidad de Castilla-La Mancha. 2015. URL: http://visilab.etsii.uclm.es/?page_id=1468 (visitado 12-02-2017).

[52] Gilbert Strang. «The discrete cosine transform». En: *SIAM review* 41.1 (1999), págs. 135-147.

[53] Dante I Tapia y Juan M Corchado. «An ambient intelligence based multi-agent system for alzheimer health care». En: *International Journal of Ambient Computing and Intelligence (IJACI)* 1.1 (2009), págs. 15-26.

[54] Farzaneh Tatari, Mohammad-R Akbarzadeh-T y Ahmad Sabahi. «Fuzzy-probabilistic multi agent system for breast cancer risk assessment and insurance premium assignment». En: *Journal of Biomedical informatics* 45.6 (2012), págs. 1021-1034.

[55] Antonio Tejero de Pablos y col. «Sistema de reconocimiento de acciones mediante cámaras con detección de profundidad». En: (2013).

[56] Paolo Tormene y col. «Matching incomplete time series with dynamic time warping: an algorithm and an application to post-stroke rehabilitation». En: *Artificial intelligence in medicine* 45.1 (2009), págs. 11-34.

[57] Michael E Wall, Andreas Rechtsteiner y Luis M Rocha. «Singular value decomposition and principal component analysis». En: *A practical approach to microarray data analysis*. Springer, 2003, págs. 91-109.

[58] Xiaoyue Wang y col. «Experimental comparison of representation methods and distance measures for time series data». En: *Data Mining and Knowledge Discovery* 26.2 (2013), págs. 275-309.

[59] WikiMedia. *LaTeX Wikibook*. 2010. URL: http://en.wikibooks.org/wiki/LaTeX (visitado 02-02-2017).

[60] Yaodong Zhang y James Glass. «A piecewise aggregate approximation lower-bound estimate for posteriorgram-based dynamic time warping». En: *Twelfth Annual Conference of the International Speech Communication Association*. 2011.