Universidad de Valladolid

UNIVERSIDAD DE VALLADOLID

E.T.S.I. TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA DE TECNOLOGÍAS ESPECÍFICAS DE TELECOMUNICACIÓN. MENCIÓN EN TELEMÁTICA

Sistema de tele-rehabilitación a domicilio con chatbot y microordenador de bajo coste

Autor:
D. Javier Delgado Rodríguez
Tutor:
D. Mario Martínez Zarzuela y Dña. Cristina Simón Martínez
Valladolid, 23 de agosto de 2021

Universidad de Valladolid

Agradecimientos

En primer lugar, a Mario y a Cristina, agradecer el esfuerzo empleado y la oportunidad que me habéis dado de participar en este bonito proyecto. Espero haberos devuelto esa confianza con el esfuerzo empleado en desarrollar la aplicación. Sin olvidarme de Beatriz, que gracias a ella he obtenido el punto de vista clínico necesario para el desarrollo del proyecto.

Como punto y final a esta etapa, agradecer a mis padres y a mi hermana el esfuerzo y confianza puesta en mí desde que comencé este viaje hace cinco años, sin vosotros no hubiera llegado a donde estoy.

Y a Estela. Por tu apoyo incondicional durante estos años. Lo mereces todo y más.

Universidad de Valladolid

TÍTULO:	Sistema de tele-rehabilitación a domicilio con chatbot y microordenador de bajo coste
AUTOR:	D. Javier Delgado Rodríguez
TUTOR:	D. Mario Martínez Zarzuela Dña. Cristina Simón Martínez
DEPARTAMENTO:	Departamento de Teoría de la Señal y Comunicaciones e Ingeniería Telemática

TRIBUNAL

PRESIDENTE:	MÍRIAM ANTÓN RODRÍGUEZ
VOCAL:	DAVID GONZÁLEZ ORTEGA
SECRETARIO	MARIO MARTÍNEZ ZARZUELA
SUPLENTE	FRANCISCO JAVIER DÍEZ PERNAS
SUPLENTE	CARLOS GÓMEZ PEÑA

FECHA:
CALIFICACIÓN:

Resumen

El trabajo desarrollado ha tenido como objetivo encontrar un sistema simple, de bajo coste y organizado que permita a los pacientes rehabilitarse desde casa manteniendo el sentimiento de cercanía por parte de los terapeutas en las rehabilitaciones presenciales.

Para cumplir con el objetivo se ha desarrollado un sistema que hace uso de tecnologías de bajo coste, como la Raspberry Pi, para el desarrollo de un bot en Telegram haciendo uso de la Application Programming Interfaces (API) de Telegram y el desarrollo de una aplicación web, con la que se consigue crear un canal de comunicación entre terapeuta-paciente. Se busca mantener los mismos aspectos sociales y la misma motivación que a través de las rehabilitaciones presenciales.

Durante los próximos apartados de este documento, se podrán ver detallados cada uno de los pasos que se han seguido durante el desarrollo de este sistema hasta que se ha conseguido el resultado final. Finalmente, se incluyen unas breves conclusiones y las diferentes líneas de trabajo futuro.

Palabras clave: tele-rehabilitación, *framework*, computación en la nube, aplicación web, bot.

Abstract

The project carried out has aimed to find a simple, low-cost and organized system that allows patients to rehabilitate from home while maintaining the feeling of closeness on the part of therapists in face-to-face rehabilitation.

To reach the objective, a system has been developed that makes use of low-cost technologies, such as the Raspberry Pi, for the development of a bot in Telegram using the Telegram API and the development of a web application, with which it is possible to create a communication channel between therapist-patient.

It seeks to keep the same social aspects and the same motivation as through face-to-face rehabilitation.

During the next sections of this document, each of the steps that have been followed during the development of this system will be detailed until the final result has been achieved. Finally, some brief conclusions and the different lines of future work are included.

Keywords: *tele-rehabilitation*, *framework*, *cloud computing*, *web application*, *bot*.

Abreviaturas

AWS: Amazon Web Services

API: Application Programming Interfaces

BBDD: Bases de datos

MTV: Model Template View

Universidad de Valladolid

Índice de contenido

Universidad de Valladolid

Índice de imágenes

Universidad de Valladolid

Índice de tablas

1. Introducción

La inclusión de las nuevas tecnologías en nuestra vida también ha supuesto una transformación en la medicina. A lo largo de los últimos años, factores como el elevado coste del material, la dificultad para acudir a las citas médicas y la deficiente participación del paciente ha provocado que la rehabilitación convencional busque un hueco dentro de la digitalización. La tele-rehabilitación se define como el conjunto de servicios de rehabilitación con la ayuda de la tecnología informática y la comunicación. Gracias a ella, se consigue disminuir los factores de la rehabilitación presencial y proporcionar una rehabilitación personalizada a distancia manteniendo la comunicación entre el terapeuta y el paciente (1).

Las terapias con tele-rehabilitación han resultado ser eficaces, flexibles y personalizables reforzando que este tipo de tecnología es una alternativa valiosa para los tratamientos de rehabilitación. Además, se podría considerar como una estrategia útil para pacientes que sufren un trastorno neurológico como por ejemplo en niños con parálisis cerebral (2).

La situación generada por la pandemia de la Covid-19 ha acelerado el proceso de digitalización del planeta. Durante la pandemia se ha paralizado el acceso de los niños a las terapias de rehabilitación. Como consecuencia han ido apareciendo en los niños síntomas de estrés infantil, disminución de la movilidad y reducción del ejercicio físico (3). La pandemia ha reducido nuestra vida social, algo tan necesario en actividades como la rehabilitación, lo que ha supuesto que se haya comenzado a valorar la posibilidad de compatibilizar las terapias presenciales con las no presenciales haciendo uso de la tecnología. Lo que se pretende conseguir es un sistema que integre todo lo necesario para que pacientes y terapeutas obtengan la misma eficiencia del tratamiento, pero de forma no presencial (3).

Gracias a las nuevas tecnologías podemos obtener más datos sobre la evolución de los pacientes optimizando su rehabilitación al máximo. Con la ayuda de estas herramientas se puede tener un mayor control del progreso de los pacientes, de tal manera que podemos ser capaces de informatizar y establecer modelos que evalúen el progreso de los pacientes.

La tecnología y la medicina unidas podrían servir para mejorar la vida de las personas, aportando datos necesarios para monitorizar los tratamientos que se individualizan para cada uno de los pacientes. Para ello, es necesario encontrar una tecnología que reúnan tres características indispensables: tener un bajo coste, ser fácil de usar y estar organizado de tal manera que pacientes y terapeutas mantengan los aspectos sociales y la misma motivación que en la rehabilitación presencial. De esta manera, se podría mantener una relación personalizada con los pacientes y sus familias de una forma similar que cuando se acuden a las terapias presenciales.

1.1. Contexto y motivación

Este proyecto surge con la idea de mejorar las funciones motoras de pacientes con parálisis cerebral incluyendo la tele-rehabilitación en su tratamiento. En concreto, nos centraremos en niños diagnosticados con parálisis cerebral infantil unilateral de entre dos y doce años. La parálisis cerebral es un conjunto de alteraciones permanentes que afectan tanto a la movilidad como a la postura y que se desarrolla durante el crecimiento del cerebro de los neonatos (4). A la hora de clasificar la parálisis cerebral las más usadas han sido las

propuestas por Ingram y Hagberg. Por un lado, Ingram clasifica la parálisis cerebral según el tipo clínico en diplejía, hemiplejía, tetraplejía, ataxia, discinesia y tipos mixtos. Mientras que la clasificación de Hagberg, se divide en síndromes espásticos, síndromes extrapiramidales (discinéticos) y ataxia (4).

Desde un primer momento se busca una tecnología que aporte beneficios que de forma convencional sean más difíciles de conseguir. Como, por ejemplo, la recopilación de datos con sensores vestibles en el cuerpo o la automatización de las sesiones para que los pacientes hagan rehabilitación desde sus hogares. Siempre con la vista puesta en compatibilizar las sesiones presenciales y las no presenciales para beneficiarnos de ambas situaciones.

La COVID-19 ha supuesto la cancelación de muchos programas activos dentro de la atención primaria, incluidos la rehabilitación. Por lo tanto, ha crecido la necesidad de buscar la forma de mantener el sistema público de salud activo en todos sus campos que no son de urgencia, es decir, lograr solucionar las actividades no urgentes en cualquier tipo de situación que pueda provocar una alteración en el habitual funcionamiento de un centro hospitalario, tanto privado como público. El proceso de digitalización de la sanidad pública puede llegar a garantizar los servicios no urgentes en situaciones adversas, y en este caso a través de la tele-rehabilitación, conseguir mantener los tratamientos activos para los pacientes.

Los acontecimientos vividos en estos años deben servir de aliciente para tratar de incluir nuevas herramientas en los tratamientos y continuar con la digitalización de la sanidad pública. Gracias al tipo de tecnología que se usa en este proyecto, tenemos un sistema escalable que puede evolucionar y ampliar sus miras hacia diferentes tipos de diagnósticos, buscando la manera de mejorar la calidad de vida de las personas que sufren algún tipo de patología grave.

1.2. Objetivos

El objetivo principal de este **Trabajo de Fin de Grado** es el desarrollo de un sistema que automatice el proceso de tele-rehabilitación de pacientes diagnosticados con parálisis cerebral. Asimismo, se abordarán los siguientes objetivos secundarios:

1. Las tecnologías usadas en este proyecto deben ser asequibles para todos los pacientes diagnosticados de parálisis cerebral.
2. El sistema final debe ser fácil de usar tanto para los terapeutas, que harán uso de la aplicación web, como para los pacientes que utilizarán el bot.
3. El sistema de tele-rehabilitación debe ser organizado manteniendo coordinados a terapeutas, pacientes y padres en el proceso de rehabilitación desde casa.

Para abordar los objetivos se ha desarrollado una aplicación web para que los terapeutas puedan mantener un seguimiento de la evolución de sus pacientes. Además de ello, los terapeutas van a poder crear y editar, desde un panel de administración, a los usuarios, los ejercicios y las sesiones de forma sencilla.

Por otro lado, el paciente recibirá un sistema compuesto por una Raspberry Pi Model 4B y una pantalla LCD de 7 pulgadas. Con la Raspberry Pi conseguiremos sincronizar las terapias previamente generadas por los terapeutas y enviárselas a los padres y/o tutores a través de un bot en Telegram. El bot será el encargado de organizar las terapias, enviar los recordatorios a los pacientes de las sesiones y finalmente recogerá la retroalimentación de

los padres y/o tutores sobre los ejercicios realizados por los pacientes, así como los problemas que hayan podido surgir a lo largo de la sesión. La pantalla creará un ambiente con mayor dinamismo en el que los niños podrán ver los videos con los ejercicios que tienen que realizar con ayuda de sus padres y/o tutores. La finalidad de la pantalla es conseguir que la sesión de rehabilitación sea más interactiva y mantenga a los niños centrados en completar los ejercicios.

Más adelante detallaremos los requisitos que deben cumplir las partes que conforman el sistema.

1.3. Fases y métodos

Cuando nos enfrentamos a un proyecto que requiere el desarrollo de aplicaciones o de *software*, podemos usar diferentes modelos para la planificación, el desarrollo y la implementación de la aplicación. Por un lado, disponemos de los modelos tradicionales o también conocidos como clásicos. Algunos de los más usados son los siguientes:

- **Modelo en Cascada**: las fases del proyecto se van planificando una tras otra. Una fase no comienza hasta que su antecesora no está completa. Este modelo tiene sus propias versiones y una de ellas es la que propuso Winston W. Royce.

Como se observa en la **imagen 1**, esta versión hace uso de 5 niveles. El primer nivel corresponde a la fase de análisis. Royce presenta un modelo en el que cada paso a la siguiente fase requiere de una comprobación de los resultados obtenidos con los que en un primer momento se ha solicitado. Además, en la fase de análisis se incluye la planificación y las especificaciones de los requisitos. El segundo nivel, es la fase de diseño. En esta fase, nos enfrentaríamos al diseño de la estructura del sistema. El tercer nivel, la fase de implementación, en la que el desarrollador debe de completar la programación y realizar algunas pruebas individuales. El cuarto nivel consta de la fase de verificación del sistema que integra la funcionalidad del proyecto. Es muy importante comprobar que el sistema está funcionando de forma correcta. Y el ultimo nivel, la fase de mantenimiento.

Hay que tener en cuenta que cualquier sistema que se desarrolle va a necesitar un mantenimiento. Por esta razón, es necesario mantener el sistema actualizado y corregir los posibles errores que se vayan produciendo, así como, ir incluyendo las mejoras pertinentes con el paso del tiempo.

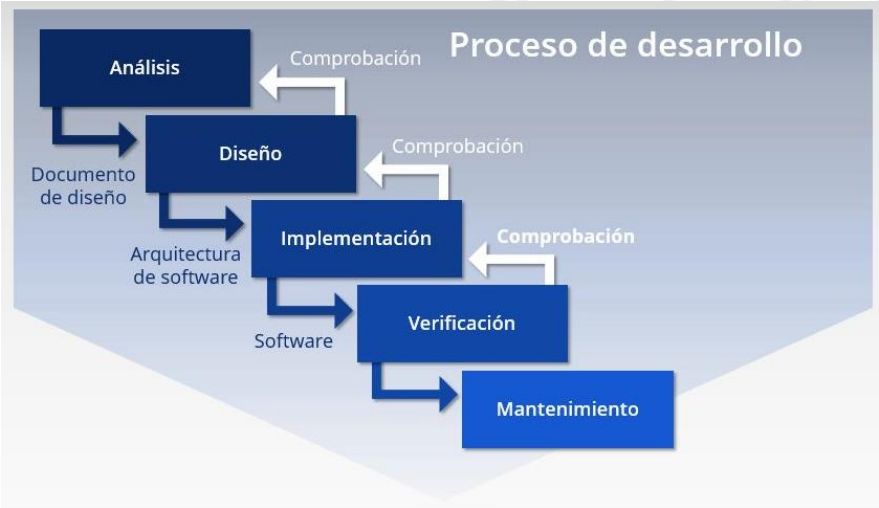

Imagen 1 . Estructura del Modelo en Cascada en la versión de Winston W. Royce.

- **Modelo basado en Prototipos o Iterativo**: es la versión evolucionada del modelo en cascada. Este modelo permite que el cliente vaya descubriendo lo que quiere según va viendo pequeñas demostraciones. Para ello, tienen ciertas premisas que cumplir:
  1. Conseguir un prototipo cuanto antes, para que el cliente lo vaya validando.
  2. Ir evolucionando las funcionalidades en cada uno de los ciclos. De esta forma el cliente podrá ir viendo los beneficios que le puede ofrecer el sistema que se está desarrollando.

Es muy beneficioso para el cliente obtener resultados de forma rápida. Además, el constante *feedback*, entre ambas partes, permite adaptarse rápidamente a las necesidades del sistema. El proyecto se va desarrollando en pequeños trozos en los que se va adquiriendo, poco a poco, el conocimiento sobre el producto final.

El inconveniente de este modelo es la necesidad de tener un cliente que se involucre en el desarrollo del *software*. Con este modelo es muy probable que las especificaciones vayan cambiando y las conversaciones deben ser fluidas para el beneficio de ambas partes.

Al igual que en el modelo en cascada teníamos diferentes versiones del modelo, en el iterativo también tenemos, es el caso del modelo iterativo e incremental.

Como podemos ver en la **imagen 2**, la principal diferencia es que en cada entrega aparte de tener funcionalidades nuevas (incremental), se incluyen mejoras en lo que ya estaba desarrollado (iterativo).

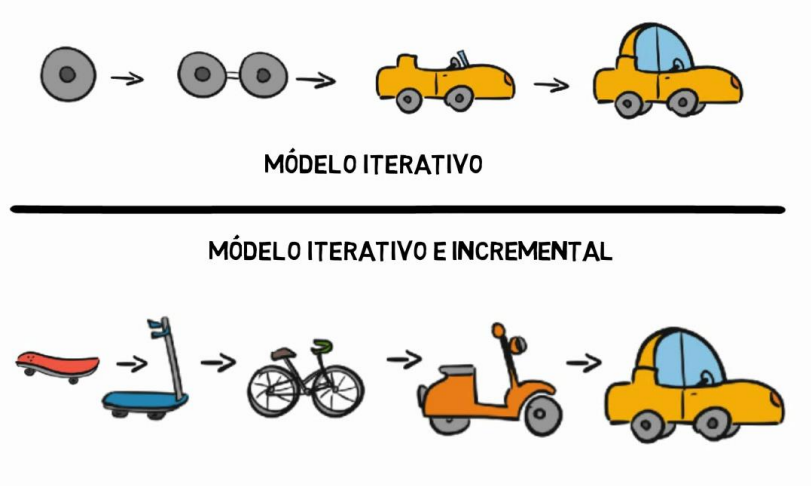

Imagen 2. Diferencias entre el modelo iterativo y el modelo iterativo e incremental.

- **Modelo en Espiral**: es la combinación de los dos anteriores. Se destina a proyectos donde existe un gran riesgo a cometer fallos y en definitiva se va evaluando los posibles riesgos que existen y aumentando poco a poco las funcionalidades del *software*. Los proyectos suelen ser de larga duración, complejos y requieren un gran presupuesto.

En la **imagen 3**, se puede observar las fases del modelo en espiral. Cada una de las fases de las que se compone, es más duradera, más compleja y requiere una mayor planificación.

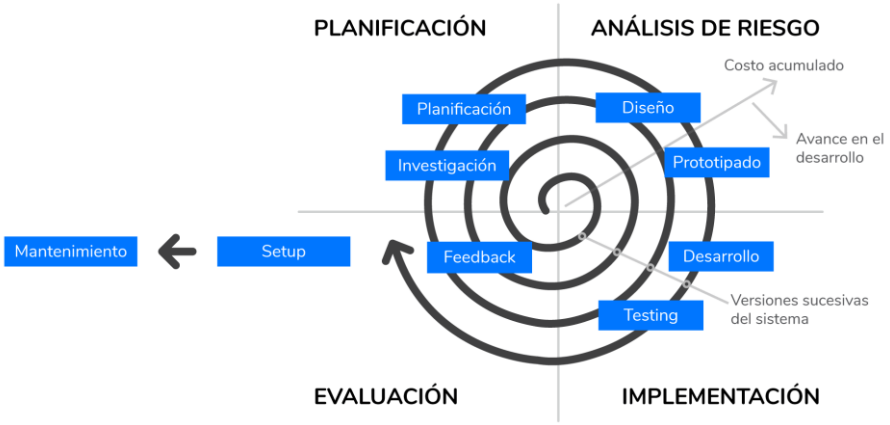

Imagen 3. Estructura del modelo en espiral.

Existen infinidad de modelos que se utilizan para el desarrollo *software*. Además de los modelos que hemos catalogado como clásicos tenemos muchos otros, entre ellos destaca el **modelo ágil**. Este tipo de modelos surgen en los años 90 y se han ido desarrollando hasta nuestros días, llegando a sentar las bases, por las que en la actualidad se rige (5). Estas bases fueron sentadas por expertos en las que declaran (5):

1. La satisfacción del cliente es la prioridad máxima.
2. Los cambios son bienvenidos en cualquier momento del desarrollo ya que nos permiten ser más competitivos.
3. Es necesario presentar el *software* en funcionamiento en períodos cortos de tiempo.
4. Se debe trabajar codo con codo con los clientes. Preferiblemente mantener el contacto cara a cara.
5. Es de gran importancia ser organizados y maximizar la eficiencia durante el desarrollo.
6. Se debe reflexionar de vez en cuando las mejores formas de continuar con el proyecto.

Una vez descritas las principales características de estos modelos, es importante destacar que para el desarrollo de este proyecto se ha seguido un modelo muy similar al iterativo e incremental. Por lo tanto, este proyecto se ha regido por un método clásico aprovechando las continuas reuniones con los participantes de este proyecto.

El proyecto ha tenido una fase inicial en la que se ha planificado *a grosso modo* las funcionalidades que debía de tener la aplicación. Sin ser específicos y sin conocer el tiempo necesario para el desarrollo e implementación de la aplicación se han fijado los objetivos principales que ya hemos comentado con anterioridad y que más adelante veremos en el apartado de requisitos del sistema.

En cada reunión con el equipo del proyecto, se han revisado los objetivos logrados hasta el momento y como muy bien se aprecia en la **imagen 3**, en cada reunión se iban definiendo especificaciones nuevas que debía de tener la aplicación. En definitiva, el proyecto ha ido

creciendo exponencialmente una vez se han ido logrando versiones del sistema y se ha ido conociendo la potencialidad de este.

Hay que destacar la importancia del constante contacto con el equipo de trabajo y gracias a las herramientas que se han usado en el proyecto pronto se han ido mostrando avances y ajustando las características que este debía de tener.

1.4. Recursos *hardware y software*

Para la realización de este proyecto ha sido necesario tanto el uso de *software* como de *hardware*. Por un lado, tenemos la parte *software*, en la que se ha necesitado hacer uso de herramientas específicas para el despliegue de la aplicación. Una de ellas, la más importante, ha sido *Amazon Web Services* (AWS) que nos ha proporcionado un servidor en el que va a estar ejecutándose la aplicación. Además, interviene en otra de las partes clave del proyecto, el almacenamiento de los datos. Entre los muchos datos que son necesarios recopilar para el correcto funcionamiento de la aplicación tenemos que destacar:

- Videos con los ejercicios que los pacientes van a realizar
- Ficheros de retroalimentación
- Ficheros con las sesiones de los pacientes
- Copia de seguridad de las bases de datos alojada en el servidor
- Ficheros requeridos por el Bot para su correcto funcionamiento

Por lo tanto, desde AWS se mantiene activo toda la aplicación y hace de unión entre los muchos sistemas que esta interconectados entre sí. Para que todo ello funcione en sintonía se han usado diferentes tecnologías, como, por ejemplo, sistemas de gestión de base de datos, *framework* para la aplicación web, un servidor web, etc. Todo ello integrado dentro del servidor de Amazon, ofreciéndonos una gran escalabilidad a largo plazo e infinidad de servicios. El conjunto de este proyecto no ha sido únicamente *software*, sino que se ha hecho uso de diferente *hardware* para el completo funcionamiento del sistema. Por un lado, tenemos una Raspberry Pi que hace de unión entre el servidor de AWS y los usuarios finales del sistema, los pacientes. El modelo necesario para el desarrollo de este proyecto ha sido:

- Raspberry Pi 4 Model B+ con 8 GB de RAM
- Tarjeta micro-SD de 32GB
- Caja oficial para Raspberry Pi 4
- Fuente de alimentación de 5.1V y 3ª

2. Estado del arte

2.1. Trabajos previos

Antes de centrarnos en nuestro proyecto es interesante hacer un breve repaso de tecnologías ya existentes que se aplican o se han aplicado para la tele-rehabilitación y cuáles han sido sus resultados. Para ello vamos a revisar varios estudios que se centran en la rehabilitación domiciliaria a través del uso de diferentes tecnologías y metodologías.

En uno de los estudios revisados, detallan los beneficios de la realidad virtual aplicada a la tele-rehabilitación, con el fin de mejorar la movilidad y el equilibrio de pacientes que han sufrido un accidente cerebrovascular. En este estudio participaron seis pacientes durante 3 semanas, dos en la clínica y una en casa. El proyecto consistía en un bastidor de bipedestación para el entrenamiento del equilibrio. Se configuraban diferentes escenarios en el que el paciente iba equilibrando su cuerpo en función de los obstáculos que iban apareciendo en las diferentes escenas que se le iban presentando en la pantalla. Los resultados obtenidos en este estudio reflejaban que el uso de la realidad virtual en la tele-rehabilitación mejoraba el equilibrio y la postura en estos pacientes. Se concluyó que el entrenamiento en el hogar tenía un resultado similar al que se obtenía en la clínica, pero este nuevo enfoque permite reducir las visitas a la clínica, aporta mayor autonomía al paciente, reduce costes y permite tratar a más pacientes simultáneamente (6).

Una prueba de otro sistema de tele-rehabilitación es lo que podemos observar en el siguiente estudio. En el estudio participaron 12 pacientes que habían sufrido un accidente cerebrovascular. Cada uno de ellos recibió durante 28 días un programa de tele-rehabilitación en su hogar basado en diferentes juegos y ejercicios. Todos ellos se basaban en actividades dinámicas y enfocadas en los principios de plasticidad y el aprendizaje. El programa de rehabilitación consistía en la realización de los ejercicios que previamente el terapeuta había seleccionado de forma personalizada. Dentro de la hora programada para el entrenamiento además de realizar las actividades, cada tres veces por semana se hacían videoconferencias con los terapeutas en las que se realizaba un seguimiento con el paciente de cada una de las actividades realizadas. El sistema se componía de una mesa, una silla, un ordenador, una cámara web interna, un módem USB inalámbrico, un brazalete de presión arterial y una alfombrilla USB con interruptores. Este estudio evidencia los beneficios de mantener los programas de rehabilitación también desde el hogar de los pacientes (1).

En este siguiente estudio se describe la inclusión de una Nintendo Wii en el programa de rehabilitación en casa en el que participaron niños entre 7 y 12 años con parálisis cerebral hemipléjica espástica. El uso de la Wii aportaba a este tipo de pacientes la motivación necesaria para completar el entrenamiento, aprovechando este hecho para, a su vez, aumentar la función motora de los pacientes. El entrenamiento con la Wii ha resultado ser eficaz para mejorar las capacidades motoras en estos pacientes y podría ser un punto de partida para la inclusión de videojuegos como terapia. Sin embargo se requieren más investigaciones para reafirmar y completar la aplicación de este tipo de rehabilitación (7).

De estos estudios se puede observar la necesidad de implementar programas de rehabilitación adaptados a cada situación y a las necesidades de los pacientes. Además, resulta necesario adaptarse a los cambios sociales y evolucionar con los avances tecnológicos y los beneficios que nos pueden aportar. Sin embargo, son muchos los estudios que se tienen que realizar para determinar que patologías clínicas se pueden beneficiar de ello.

2.2. Revisión de tecnologías

En este apartado haremos una revisión de la evolución de las tecnologías usadas para el desarrollo de este sistema a lo largo de los años. Nos centraremos en las herramientas que hemos usado en este proyecto haciendo hincapié en las características principales que las diferencian de las demás.

2.2.1. Desarrollo web - *Frameworks*

A lo largo de los últimos 30 años el desarrollo web ha experimentado una gran revolución en todos los aspectos. Las primeras páginas web de las que tenemos conocimiento nacieron en la década de los años 90. Por aquel entonces, las páginas tenían una pantalla negra con colores monocromáticos. En realidad, fue en el año 1991 cuando se publicó la primera página web (WWW) de la mano de uno de los considerados padres de la Web, Tim Berners-Lee.

En la actualidad el diseño web se encuentra en auge, ya no es necesario tener conocimientos avanzados en diseño web, sino que disponemos de herramientas que nos permiten crear sitios web de una forma increíblemente sencilla. Si es verdad que con el paso del tiempo los requerimientos de las páginas web van creciendo y con ello las herramientas que tenemos a nuestra disposición.

Cuando hablamos de páginas web podemos dividirlas en dos partes, el *front-end* y el *back-end*. Cuando hablamos de *front-end* nos referimos a la parte con las que los usuarios interactuamos. La estética y la disposición de los elementos en una página web es muy importante, ya que debe de ser atractiva para los usuarios. Y no solo eso, además debe de tener algunos requisitos esenciales como la responsividad, usabilidad y accesibilidad. En muchos casos no es necesario nada más en una página web, pero en muchos otros necesitamos dotar a la página de "inteligencia".

Es en esta parte donde entra en juego el *back-end.* El *back-end* es una parte del desarrollo web que actúa de forma transparente a los usuarios. Gracias a ello podemos crear aplicaciones web que van más allá de mostrar la información organizada. La existencia de muchos *frameworks* como Django, Laravel o Flask posibilita crear aplicaciones web de cualquier tipo como, por ejemplo, aplicaciones web médicas, aplicaciones web de compraventa, etc.

En nuestro proyecto ha sido necesario el uso de uno de estos *frameworks*, es el caso de Django. El proyecto se divide en varias partes, una de ellas es la parte de la aplicación web. Para esta parte se ha creado una aplicación web que por un lado tiene la parte informativa en la que se quiere exponer el trabajo que estamos desarrollando y por el otro lado se encuentra la parte de aplicación como tal, que está configurada únicamente para usuarios con acceso a ella.

Elección del *framework* para el desarrollo de la aplicación web

La necesidad de crear una página web con inteligencia, ha provocado el estudio de las diferentes herramientas que permiten crear sitios web dinámicos, escalables y potenciales. He llevado a cabo una breve comparación entre las herramientas más usadas actualmente para el desarrollo web. Realmente muchas de ellas son muy similares, arquitecturas que varían levemente unas de otras, pero con un objetivo final conjunto que se trata de crear sitios web seguros y de una forma rápida (8). Por comentar los aspectos más relevantes de los *frameworks* más usados he querido listar los beneficios que nos ofrecen estas herramientas (8):

- Creación rápida de una página web
- Seguridad y fiabilidad
- Escalabilidad
- Rendimiento

Si bien es cierto, no todas usan el mismo lenguaje y eso afecta en muchas ocasiones al rendimiento, rapidez o a la fiabilidad. Por ello antes de elegir una u otra es importante determinar la funcionalidad del sitio web y sus necesidades. Por ejemplo, Django se suele destinar a aplicaciones que requieran un cierto grado de seguridad, ya que el sistema de autenticación de usuarios del que dispone es más potente que otras (8). Por otro lado, Laravel funciona muy bien cuando se quiere crear un sitio web para la transmisión de video en directo o comunidades en línea.

La comparación entre Django y Lavarel, se puede observar en la **imagen 4.** Aunque Laravel lidera el número de sitios web en los que se hace uso de ella, su tendencia está decreciendo en comparación con Django. Se debe en gran parte a que Django usa un lenguaje de alto nivel como es Python, haciendo hincapié en la búsqueda del desarrollo rápido y el diseño limpio. Además, con Python obtenemos una mayor flexibilidad, escalabilidad y versatilidad que con PHP que es el marco de trabajo que usa Laravel (9).

Imagen 4. Comparativa Django y Laravel (9).

Como podemos imaginarnos el catálogo de herramientas es inmenso y no para de crecer según las necesidades del mercado que van surgiendo.

La elección de usar Django en este proyecto no se ha basado en un comparativa como la que nos muestra la imagen 4. La finalidad de este proyecto, las facilidades que ofrece Django (como el panel de administración por defecto o la compatibilidad con gran variedad de gestores de bases de datos), el uso de Python como lenguaje de programación, el sistema de autenticación, el rápido despliegue y el número de *plugins* que ofrecen, decantaron la balanza en un mercado lleno de posibilidades.

Arquitectura de Django

El primer paso para entender el funcionamiento de Django es comprender su arquitectura, puesto que rompe con las arquitecturas convencionales conocidas. Por una parte, tenemos el modelo (M), que sería la parte que mantiene unida la aplicación con la base de datos (acceso, validación, relaciones entre los datos…). Después tendríamos las plantillas (T), donde se diseña y configura la presentación de cada una de las páginas web, es decir, lo que el usuario final va a percibir. Finalmente, tenemos la vista (V) que contiene toda la lógica de negocio de la aplicación web y hace de puente entre las vistas y las plantillas. Todo unido da lugar a la arquitectura *Model Template View* (MTV) (10).

En la **imagen 5**, se puede observar que existen tres capas que son independientes pero que se unen en una solo, la aplicación web final. Además, el usuario sólo percibe la vista, es decir, lo que el programador ha diseñado para mostrar a los usuarios en cada una de las secciones de la página web, haciendo uso de las plantillas para mostrar la información. Para el usuario es irrelevante lo que existe por debajo de la vista, ya que sólo percibe el funcionamiento de la página.

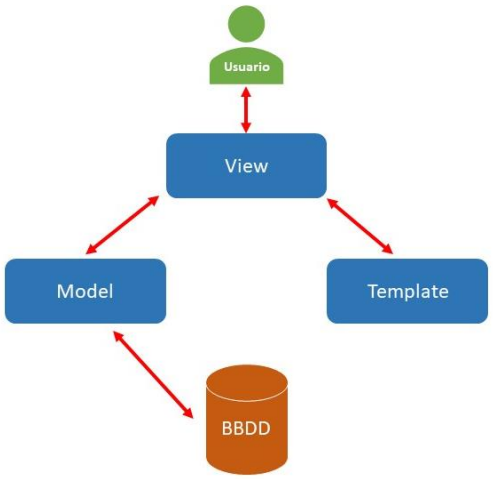

Imagen 5. Esquema de la arquitectura MTV.

Las comunicaciones entre las distintas capas son de la siguiente forma, un usuario accede a la página y se encuentra con lo que el programador ha definido en la vista y lo que finalmente se muestra en la plantilla (Template). Cuando una de las vistas requiere hacer una consulta a una base de datos (BBDD) lo que sucede es que se hace la petición a una o varias tablas definidas en el modelo. La base de datos devolverá una serie de objeto dependiendo de la

consulta realizada, la vista lo procesará y, por último, se mostrará a través de la plantilla definida para esa sección (10).

Características principales

Una vez descrita la arquitectura de Django, podemos definir los aspectos más relevantes que le convierten en una herramienta para el desarrollo web (11).

1. Escalabilidad, permite crear sitios web dinámicos que se adaptan rápidamente a las necesidades de la aplicación.
2. Seguridad, incluye un sistema de autenticación de usuarios muy seguro, evita errores en las BBDD y falsificación de solicitudes, etc.
3. Documentación extensa, el proyecto de Django incluye diferentes versiones, con su respectiva API, widgets y continuas actualizaciones.
4. El uso de la arquitectura MTV permite un desarrollo ágil y reutilizable.
5. Cuenta, por defecto, con un panel de administración que permite el control simplificado de las bases de datos y permite añadir diferentes *plugings* que mejoran la experiencia de usuario.
6. Uso de Python del que hereda todas las facilidades que nos ofrece este lenguaje, incluyendo la simplicidad de generar aplicaciones fáciles de entender.

2.2.2. Sistemas de chatbot

En la actualidad el número de aplicaciones que permiten la creación de bot aumentan día a día. Podemos destacar aplicaciones como Telegram, Facebook, PandoraBots, Signal, Whatsapp, etc. Para estudiar el funcionamiento de estas herramientas vamos a centrarnos en algunas de las más importantes, es el caso de: Telegram, Chatfuel y Signal.

Antes de comenzar la descripción de cada una de ellas y su funcionamiento, me gustaría explicar el significado de bot. Cuando oímos la palabra bot lo primero que le asociamos es una máquina o robot que nos habla. Pero el concepto va mucho más allá de todo eso. En realidad, un bot es un programa informático que se encarga de realizar tareas de forma similar a un humano. Es decir, los bot poseen cierta inteligencia que les permite realizar infinidad de tareas, y en muchos de los casos esas tareas son una respuesta a un estímulo, reconocimiento de voz o texto de una o varias personas.

Como ya nos podemos imaginar un bot puede hacer todo lo que sea posible programar, es decir, podemos hacer lo que queramos con ellos, desde cosas buenas como cosas no tan buenas. Y en este apartado quiero destacar esta clasificación que le damos a los bot, ya que todo lo bueno que pueden ser los bots lo pueden ser de malos, dependiendo la finalidad que tengan los creadores. Podemos tener un bot monitorizando nuestra web, enviándonos informes de rendimiento o visitas de nuestra página. Pero pueden también existir bots que se encarguen de realizar ataques a un sistema, uno de los más frecuentes es el ataque por denegación de servicio (DDoS) o envío de SPAM.

En resumen, el uso de bot o la creación de estos, busca facilitarnos la vida, estando las 24 horas a nuestra disposición y realizando tareas por nosotros. Sencillamente, nosotros somos los que vamos a definir las funciones que queremos que efectúe nuestro bot, tanto las buenas como las malas, ya que los bots, por sí solos, no son ni buenos ni malos.

Una vez descrito el termino de bot, es importante conocer algunas de las herramientas que existen y comparar las características de cada una de ellas.

Telegram Bot API

En 2015, Telegram desarrolló una interfaz de programación para el uso de bot en su aplicación de mensajería. Para crear un bot es tan sencillo que basta con "hablar" por Telegram al usuario *@BotFather*. Este bot nos permite hacer diferentes cosas, algunas de ellas son las que se pueden ver en la **imagen 6** (12).

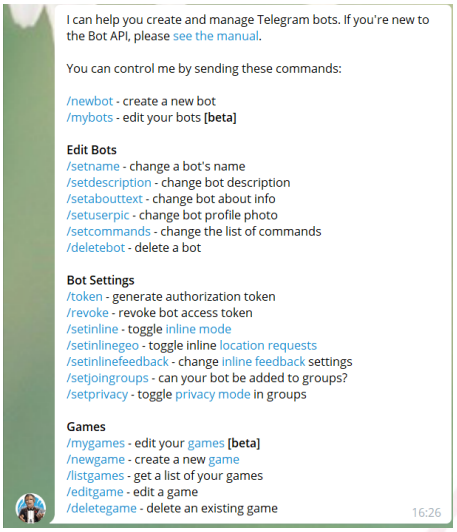

Imagen 6. Opciones disponibles en el BotFather (12).

En el caso de querer crear un nuevo bot, le debemos de mandar el siguiente mensaje: */newbot*. Con ese simple mensaje el bot padre nos envía un token con el que se identifica ese bot en Telegram. Además, se debe de añadir un nombre de usuario al bot para que se pueda encontrar en Telegram. Es importante destacar que el token es como la llave a nuestro bot y si alguien accede a esa llave puede hacer que nuestro bot envie mensajes o cualquier otra cosa. Por ello, es muy importante mantenerlo en secreto si no queremos tener problemas.

En cuanto tenemos nuestra llave, el bot está activo, pero sin ningún tipo de funcionalidad. Debemos de programar nuestro bot y para ello necesitamos hacer uso de la API de Telegram. Gracias a la excelente interfaz HTTPS de desarrollo de la que dispone Telegram, programar un Bot resulta muy sencillo.

Tenemos dos formas de representar el funcionamiento de un bot según si está destinado a un grupo o a usuarios individuales, pero el funcionamiento final es igual. Como podemos ver en la **imagen 7**, el funcionamiento es muy sencillo. Un usuario o grupo de usuarios pregunta o no, según el propósito del bot y esta contesta a lo que se le pregunta, o cumple con las funciones para las que ha sido programado.

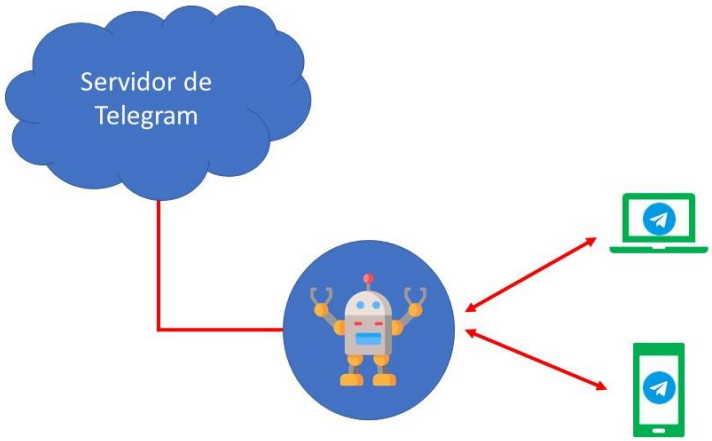

Imagen 7. Funcionamiento básico de un bot de Telegram.

Chatfuel

Para comparar las herramientas del mercado, destacamos Chatfuel como una de ellas ya que su uso es muy diferente al de Telegram. Si en Telegram el uso de la aplicación es gratuita y posee una API para crear bots para lo que necesitemos, Chatfuel, es una empresa encargada de crear bots. En primer lugar, las diferencias principales son que es de pago (pero ofrece un servicio mínimo gratis en el que permite funcionalidades limitadas) y que está orientada a empresas que quieran sacar rendimiento a los redes sociales o canales de comunicación (13).

Lo que nos ofrece esta herramienta es una aplicación capaz de crear Bot de forma simplificada, sin usar líneas de código, donde todo lo que queremos automatizar se consigue haciendo clic en una pantalla. En la **imagen 8** se puede observar cómo es la interfaz que nos ofrece Chatfuel, en este caso estamos viendo la pestaña de flujos, en ella se van creando los flujos de funcionamiento del chatbot (13).

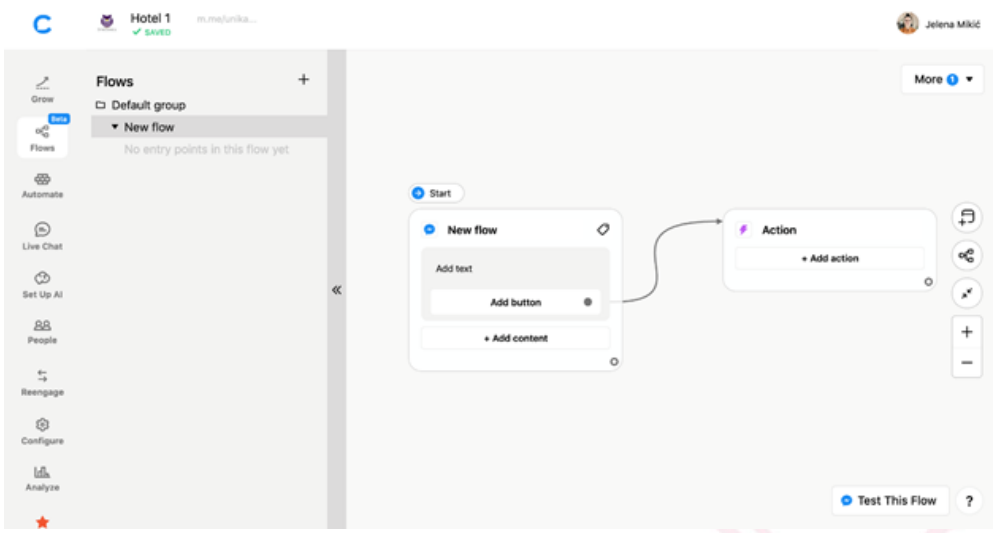

Imagen 8. Interfaz de Chatfuel (13).

Si nos adentramos un poco más dentro de la anatomía de la aplicación, veremos que el flujo del chat está formado por tarjetas que van definiendo el funcionamiento del chatbot, también permiten ejecutar alguna acción o exportar datos, entre muchas otras cosas. En la **imagen 9** podemos ver cómo es una de esas tarjetas (13).

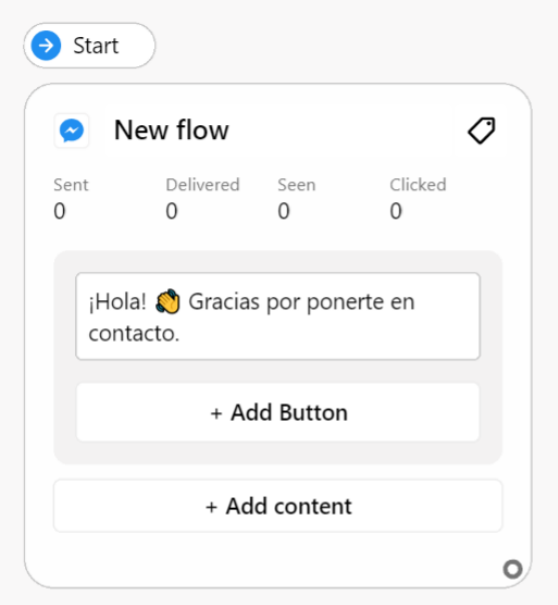

Imagen 9. Tarjeta que conforma el flujo de un chatbot en Chatfuel (13).

En resumen, las tarjetas contienen la información que el bot va a compartir con los demás usuarios y las respuestas rápidas son el conjunto de opciones que se le presentan a los usuarios para contestar a esa pregunta. Además del flujo desde Chatfuel nos permiten acceder a un panel de administración de nuestros bot donde podemos (13):

1. Crecimiento, herramientas que permiten aumentar las subscripciones de usuarios al bot de tu compañía.
2. Flujos, configuración del flujo de mensajes de los bot.
3. Automatizar.
4. Chat en directo.
5. Configurar inteligencia artificial, permite identificar las preguntas más frecuentes al bot y configurar las respuestas a ellos.
6. Personas, permite visualizar los usuarios, agrupar usuarios o generar ficheros csv para impulsar las ventas.
7. Reconectar, captar de nuevo los usuarios que alguna vez ya se han conectado.
8. Configurar.
9. Analizar.
10. Mejorar.

Como hemos podido comprobar esta herramienta y Telegram son muy diferentes, pero ambas usan la tecnología de los Bot para automatizar acciones. Si bien es cierto Telegram ofrece una API en la que puedes gestionar tus bot, programarlos y todo gratuitamente. Por

otro lado, Chatfuel ofrece a todas las personas con acceso a un teléfono o a un ordenador la manera de crear un bot sin necesidad de saber programar o incluso de lo que es un bot. Además, su mercado está expresamente dirigido a mejorar el rendimiento de las empresas, algunas de ellas son empresas de gran nombre como LEGO, Nivea, National Geographic, Adidas, Netflix, etc. Muchas de ellas en las que los bot que se han implementado se destinan a hacer recomendaciones a los usuarios, generar descuentos, ofertas y muchas aplicaciones más (13).

Signal

Signal es otra de esas aplicaciones de las que no se oye hablar mucho, pero sus funcionalidades permiten que se adapte perfectamente a las expectativas de muchos nichos de mercado. Por un lado, ofrece seguridad gracias al cifrado extremo a extremo en donde ni siquiera la propia empresa conoce el contenido de esos mensajes de texto, audios, fotos o videos que se envían a través de la aplicación. No envía publicidad a los usuarios, es decir, queda libre de spam y de seguimiento a los usuarios. Con todo eso y mucho más Signal es una aplicación gratuita que está disponible para Android, iOS, Windows o Linux (14).

En comparación con Telegram, Signal utiliza un cifrado extremo a extremo sin necesidad de servidor intermedio. En cambio, en Telegram aunque existan conversaciones privadas (no en el caso de que se use un Bot) los mensajes pasan por los servidores de Telegram. Además, Signal se distingue de sus competidores guardando únicamente los datos necesarios para la comunicación: móvil, fecha de creación y última fecha de conexión. Para acceder a la aplicación de Signal es necesario introducir un pin, de esta forma impedimos accesos no autorizados. Por último, pero no menos importante, nos ofrece crear mensajes que se autodestruyen, es decir, podemos enviar mensajes que al cabo de un rato (según el tiempo que definamos) se borran. Telegram también lo permite, pero solo en los chats privados.

Como se puede comprobar son muchas las ventajas de Signal sobre los dos competidores que hemos comentado con anterioridad, pero uno de los problemas que tiene se centra en una de las necesidades de nuestro proyecto, la creación de bots. Desafortunadamente, Signal no dispone de una API que permita crear bots con facilidad. Actualmente se encuentra en proceso de desarrollo, pero pronto veremos los primero bots con Signal.

Comparación de sistemas de chatbots

Una vez repasadas algunas de las aplicaciones más destacadas en las que se pueden crear chatbots es importante nombrar algunas de las características principales que debemos de tener en cuenta a la hora de decidirnos entre una u otra:

- **Privacidad**: el tratamiento de nuestros datos por parte de las empresas que nos ofrecen la herramienta es importante a la hora de escoger una u otra. Dentro de las anteriormente comentadas, la más destacada por su privacidad y por los métodos de seguridad que ofrece a sus usuarios es **Signal**. Por su parte **Chatfuel** y **Telegram** también nos ofrece privacidad y protegen nuestros datos, pero a un nivel inferior que **Signal**.
- **Documentación**: cuando las aplicaciones nos permiten crear con ellas otras aplicaciones específicas para cumplir los requerimientos de los usuarios, uno de los aspectos más valorables y a tener en cuenta es tener acceso a una amplia documentación, actualizada y realimentada en la que se pueda tener acceso a métodos y funciones disponibles para programar dichas aplicaciones. En este caso **Telegram**

Bot API nos lo cumple a la perfección, se encuentra actualizada, con soporte y con una gran comunidad que ofrece proyectos y ayuda con todo lo relativo a la creación de los bots.

- **Funcionalidad**: este aspecto está muy ligado al conjunto de librerías de las que dispone la aplicación. Los límites vienen marcados por los desarrolladores por ello es importante elegir una aplicación que este en continua renovación y que se actualice periódicamente. Esta característica la cumplen la mayoría de las aplicaciones que apuestan fuerte por el desarrollo de bots.
- **Requisitos de la aplicación:** el último aspecto y el más importante es definir la aplicación a la que queremos destinar nuestro bot. Un bot que envíe mensajes sencillos sin ninguna necesidad de conversación previa, un bot meramente informativo, en este caso, cualquier aplicación lo puede solventar con facilidad. Pero una vez que se va haciendo más compleja la aplicación, los requisitos cambian y es necesario buscar la herramienta que mejor se adapte a nuestras necesidades. En este caso, **Telegram Bot API** nos resuelve todas nuestras necesidades a la perfección

2.2.3. Computación en la nube

En los últimos años, la descentralización ha sido la tónica seguida por muchas empresas que ven como los hábitos los usuarios están cambiando, obligando a las empresas a modernizarse. En la actualidad esto se conoce como transformación digital, y más aún en estos meses, en los que la pandemia mundial provocada por la COVID ha acelerado este proceso mucho más. Esta transformación del modelo de negocio influye en la digitalización tanto de las PYMES como de las grandes empresas, y es en este momento donde aparece el concepto de nube. La nube es una de las revoluciones más importantes de los últimos años que permiten a las empresas ir adaptándose a las nuevas tecnologías de una forma sencilla y con un coste reducido. La idea de la nube nace de la necesidad de una reducción de costes en *hardware* y en *software* por parte de las empresas y es de esta manera, como las empresas pasan de tener su propio *hardware*, muchas veces sobredimensionado y muy costoso, a tener contratados solos los servicios necesarios para realizar su actividad empresarial (15).

Una de las ventajas más importantes que ofrecen estos servicios es la escalabilidad, flexibilidad el bajo coste y el acceso desde cualquier lugar. Aunque aún existe un gran desconocimiento de este concepto en la sociedad empresarial y en ocasiones genera reticencia entre muchos de ellos por temor a la garantía de seguridad que nos ofrece la nube (15).

Cuando hablamos de la nube o *cloud* estamos hablando de la computación en la nube o *cloud computing*. El *cloud computing* permite a un proveedor de servicios ofrecer servicios informáticos a través de internet. Entre los cuales están los servicios de almacenamiento, de redes, de servidores, de aplicaciones, etc. Un amplio abanico que permite cumplir con los requerimientos de las empresas. Para entender cómo funciona la computación en la nube tenemos que ver primero como funcionan las empresas tradicionalmente. En la **imagen 10** se puede observar la comparación entre una empresa tradicional y una empresa que utiliza cloud computing (15).

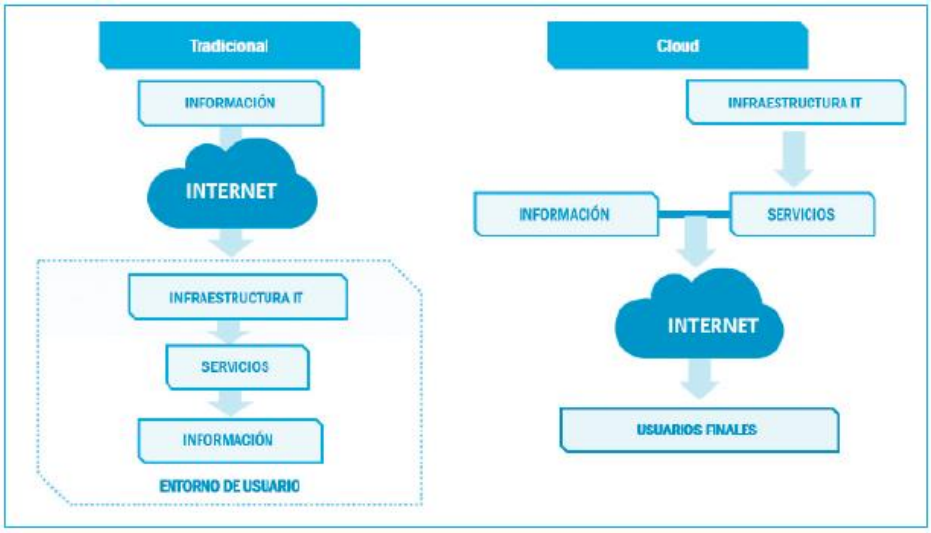

Imagen 10. Comparativa entre una empresa tradicional y una empresa que hace uso del cloud computing (15).

En la empresa tradicional, todo lo relacionado con la infraestructura IT, los servicios y la información se encuentran en el lado de los usuarios, es decir, en la empresa. Por el contrario, gracias a la nube, lo que podemos ver es como tenemos una descentralización de los servicios, la infraestructura IT y la información. Debido a ello, la empresa únicamente debe de tener acceso a internet para acceder a sus servicios, con las ventajas y los inconvenientes que eso conlleva, pero vemos como claramente se esfuma la necesidad de tener *hardware* y *software* exclusivamente para nosotros con el respetivo coste y mantenimiento que ello conlleva (15).

Entramos en un escenario en el que el empresario va a pagar por lo que necesita, va a dejar de tener que preocuparse por tener tecnología obsoleta o por tener infraestructuras sobredimensionadas. La creciente expansión en este sector se debe en parte a la expansión de los dispositivos móviles que permiten desarrollar muchas de las actividades de una empresa desde fuera de la oficina. Desde las empresas de computación en la nube es clave la facilidad de sus usuarios para acceder desde cualquier dispositivo a internet, ya que una vez que dispongamos de ello podemos acceder a nuestra oficina, virtualizada en la nube, de forma remota. Por otro lado, hay factores que han influido directamente en el cambio del modelo de negocio actual (15):

- **Capacidad de procesamiento y cálculo**: el continuo crecimiento del sector ha propiciado el desarrollo de redes de ordenadores conectados entre sí, lo que se conoce como clúster, en donde auténticas granjas de servidores unidas entre sí por una red de alta velocidad (convertidos en un único superordenador) crean sistemas con un alto rendimiento, con alta disponibilidad, con balanceadores de carga y gran escalabilidad. Los proveedores te "alquilan" un trozo de ese clúster (memoria y un número de procesadores) y el usuario paga únicamente por lo que usa.
- **Velocidad de transferencia**: gracias a los balanceadores de carga, las solicitudes se reparten de forma equilibrada entre los diferentes servidores permitiendo disponer la información de forma inmediata.
- **Expansión del acceso a Internet**: el aumento de las zonas de conectividad permite el acceso continuado a los recursos, aplicaciones o servicios.

Una de las causas que mantienen aún un distanciamiento con esta tecnología es la seguridad que nos ofrecen los proveedores de servicios en la nube. En este caso debemos de hablar de los acuerdos de nivel de seguridad (o SLA del inglés *Service Level Agreements*). Los SLA son un marco legislativo de compromisos entre el proveedor y el cliente donde se definen las responsabilidades de los proveedores en diferentes aspectos: actualizaciones, grietas de seguridad, mantenimiento, disponibilidad, etc. Después de haber caracterizado esta tecnología me gustaría repasar las ventajas y desventajas de esta tecnología (15).

Ventajas:

1. **Ahorro de costes**: los empresarios son capaces de ahorrarse la inversión en la infraestructura y en su correspondiente mantenimiento.
2. **Optimización de los recursos**: pagar por lo que se necesita. Uno de los grandes problemas en muchas empresas es afrontar el coste por equipos que están, en muchas ocasiones, sobredimensionados para la actividad que la empresa realiza.
3. **Recuperación ante desastres**: la propia descentralización de los servidores del proveedor de servicios permite que se puedan recuperar con facilidad ante una situación fatal.
4. **Actualizada y segura**: los trabajos de mantenimiento de la infraestructura es tarea de la empresa que nos ofrece los servicios, siendo en realidad, trabajos trasparentes para el usuario que hace uso del servicio.
5. **Dedicación al negocio**: gracias a la externalización de las TIC de una empresa, la empresa se puede dedicar únicamente a lo que su actividad económica le corresponde.

Aunque las ventajas son muy beneficiosas, como toda tecnología no está exenta de desventajas o puntos débiles (15).

Desventajas:

1. **Pérdida del control**: nuestros datos van a estar en manos de terceros lo que ello conduce a la desconfianza en muchos casos por no tener los datos bajo control, únicamente dependemos de los acuerdos contractuales con los proveedores del servicio
2. **Privacidad**: los datos van a estar almacenados en los servidores de los proveedores y es posible que estén expuestos a posibles ataques con el consecuente robo de los datos o eliminación de los mismos.
3. **Disponibilidad**: los servicios en la nube no están libres de problemas y cualquier momento pueden sufrir caídas que nos dejen sin servicio.
4. **Acceso a internet**: dejamos atrás el modo de conexión *offline* por lo que es un requisito indispensable tener acceso a internet para acceder a nuestros datos.

Una vez se nos ha puesto en conocimiento las características de la tecnología toca distinguir entre el tipo de servicio en la nube que queremos:

- ***Software* as a Service** (SaaS): el usuario posee la capacidad de hacer uso de aplicaciones que se están ejecutando en la nube, tales como, correo electrónico, escritorio virtual, comunicaciones, juegos, etc. El usuario dispone, sólo en algunos

casos, de la opción de configurar los parámetros de la aplicación, pero en ningún caso posee acceso de configuración a la infraestructura en la nube (15,16).

- Ventajas: reducción de costes, escalabilidad y reducción en tiempo ya que el *software* está instalado.
- Desventajas: gran dependencia del proveedor y no queda definido el dueño de la aplicación.

- **Platform as a Service** (PaaS): el consumidor es capaz de implementar una infraestructura en la nube que puede ser a través de aplicaciones propias o creadas mediante programación según el idioma, las bibliotecas o servicios que el proveedor de servicios te permita. Aplicaciones como, por ejemplo, gestión de bases de datos, herramientas de desarrollo o *big data*, entre muchas de ellas. El control del usuario aumenta hasta la aplicación y los ajustes del entorno de alojamiento de las aplicaciones. Está destinado para empresas que desean desarrollar *software* propio pero no tienen o no quiere comprar las herramientas necesarias, sino hacer uso del servicio que les ofrece una empresa externa de servicios. La seguridad en este caso está dividida entre el usuario y el proveedor (15,16).
  - Ventajas: facilidad de administración, cómoda integración con la plataforma y sencillez a la hora de generar desarrollos propios.
  - Desventajas: dependencia del proveedor y privacidad de los datos.
- **Infrastucture as a Service** (IaaS): permite al usuario procesamiento, redes, almacenamiento y más recursos con los cuales el consumidor puede implementar *software* de la manera que precise, con sistema operativo y aplicaciones (servidores, máquinas virtuales, cortafuegos, etc). El control es total por parte del usuario sobre su aplicación y todo lo que incluye, pero tampoco posee acceso a la configuración de la infraestructura de la nube. La empresa que hace uso de la nube es la encargada de mantener el *software*, pero este modelo le proporciona versatilidad a la hora de desarrollar lo que se quiera (15,16).
  - Ventajas: flexibilidad, rapidez y fácil despliegue
  - Desventajas: los problemas relacionados con el soporte del proveedor de servicios son más lentos de solucionar.

En la **imagen 11** se describe a la perfección la relación entre coste y nivel de gestión, aumentando progresivamente según vamos teniendo mayores privilegios. Por lo tanto, como ya hemos descrito en el anterior párrafo, un tipo de computación en la nube IaaS es capaz de llegar a un nivel alto de gestión, pero el precio es consecuente con la alta gestión que te permite el proveedor de servicios (15).

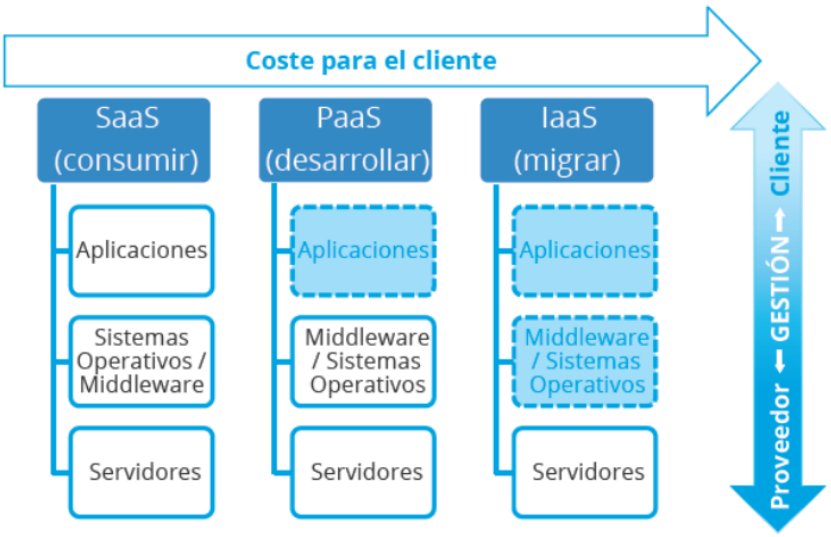

Imagen 11. Modelos de funcionamiento de la computación en la nube (15).

Los modelos de nube también se diferencian por la forma en la que se despliegan, en este caso, podemos dividirlo en tres: nube pública, nube privada y nube híbrida. Los servicios en nube pública son aquellos a los que tienen acceso muchos clientes, es decir, para un mismo servicio existen muchos clientes o varias empresa o particulares que comparten los recursos, se procesan en un mismo servidor y en ocasiones comparten espacio en el mismo disco. Por el contrario, los servicios en una nube privada mantienen los recursos aislados para cada uno de los clientes, es decir, que los recursos que se contratan son exclusivos de cada cliente. Como es obvio estoy influye en el precio del servicio, pero los clientes tienen control total sobre los recursos contratados (15,16).

Cuando combinamos ambos tipos de nube, damos lugar a la nube híbrida que combina ambos servicios, pero permite una gestión única desde un mismo panel de configuración. Se reducen costes respecto a la nube privada y logramos el máximo beneficio de los tipos de nube. Elegir entre uno u otro únicamente depende de los requisitos de nuestra empresa o necesidades propias. Cada tipo de despliegue en la nube es bueno si se destina a la tarea que está recomendada y también permiten a las empresas a ajustar los costes, ya que no siempre es posible acceder a contratar una nube privada con lo que ello conlleva (15,16).

En cuanto a la seguridad que nos ofrecen las empresas que proveen estos servicios, sabemos que ningún sistema es infalible al 100% por lo que cabe destacar y clasificar los riesgos a los que nos exponemos. En la **tabla 1**, se pueden observar las principales amenazas (dependen del tipo de servicio contratado y pueden causar daños e incidentes) y los riesgos (evalúan las amenazas) de la computación en la nube (15).

Tabla 1. Amenazas y riesgos de la computación en la nube (15).

Amenazas	Riesgos
Acceso no autorizado	Usuarios con privilegios
Amenaza interna	Incumplimiento de las normas
Interfaces inseguras	Desconocimiento de la ubicación de los datos
Fuga de información	Falta de aislamiento de los datos
Suplantación de identidad	Parada del servicio
Desconocimiento del entorno	Falta de soporte
Hackeos	Viabilidad a largo plazo

Es necesario y muy conveniente evaluar los riesgos uno a uno y mitigar sus posibles efectos, ya que no olvidemos que debemos de tratar de la misma forma los datos si los tuviéramos en local como si los tenemos en la nube.

Amazon Web Services (AWS)

AWS surgió en 2006 como empresa pionera en esta tecnología, en la que ofrecen sus propios servidores para que los clientes hagan uso de los servicios que ofrecen. Muchos de esas soluciones que ofrecen están destinadas al desarrollo de *software* por parte de los clientes, en los que se accede mediante HTTP (17).

AWS es la plataforma en nube más usada y completa que se encuentra disponible en el sector de la computación en nube. Su catálogo asciende hasta los 200 servicios disponibles en los diferentes centros de datos que poseen repartidos por el mundo. Su oferta de servicios supera la de sus competidores más cercanos, entre los que se incluyen servicios de almacenamiento, bases de datos, análisis de datos, virtualizaciones, etc. Esto hace que el despliegue de las aplicaciones en la nube sea sencillo, rápido y rentable. AWS posee una gran red de usuarios valedores de su gran potencial. Los clientes pertenecen a casi todos los sectores, independientemente de su tamaño. Tanto empresas públicas como privadas ven a AWS como la solución para la implementación de las herramientas que les permite desarrollar su actividad económica (17).

Con la idea de formar un sistema seguro pensado para bancos o el ejército fue creado AWS. Esto nos permite intuir los sistemas de seguridad que tienen implementados en sus centros de datos, ya que, con sus 230 servicios disponibles, AWS es compatible con 90 estándares de seguridad y privacidad. Además, en sus 117 servicios de almacenamiento es capaz de cifrar los datos si se desea. Como no podría ser de otra manera, es un sistema que se encuentra a la vanguardia de la tecnología e innovación, aumentando la oferta de servicios continuamente adaptándose a las necesidades de los clientes. Su experiencia está más que constatada tras 15 años prestando servicios por todo el mundo (17).

En la **imagen 12**, se puede ver como AWS es considerado el proveedor de servicios más completo de todos, un líder tanto por su infraestructura como por los servicios que nos ofrecen (18).

Imagen 12. Gartner: Evaluación de los proveedores de servicios según su infraestructura y los servicios que ofrecen, en 2020 (18).

En la **imagen 13**, se puede observar la red global de países y regiones en las que se encuentra Amazon Web Services. En ella podemos ver el despliegue realizado por todo el mundo con 80 zonas de disponibilidad que se reparten entre las 25. Además, junto a las que ya están, se han anunciado la incorporación de 15 zonas más repartidas por las regiones marcadas de amarillo (17).

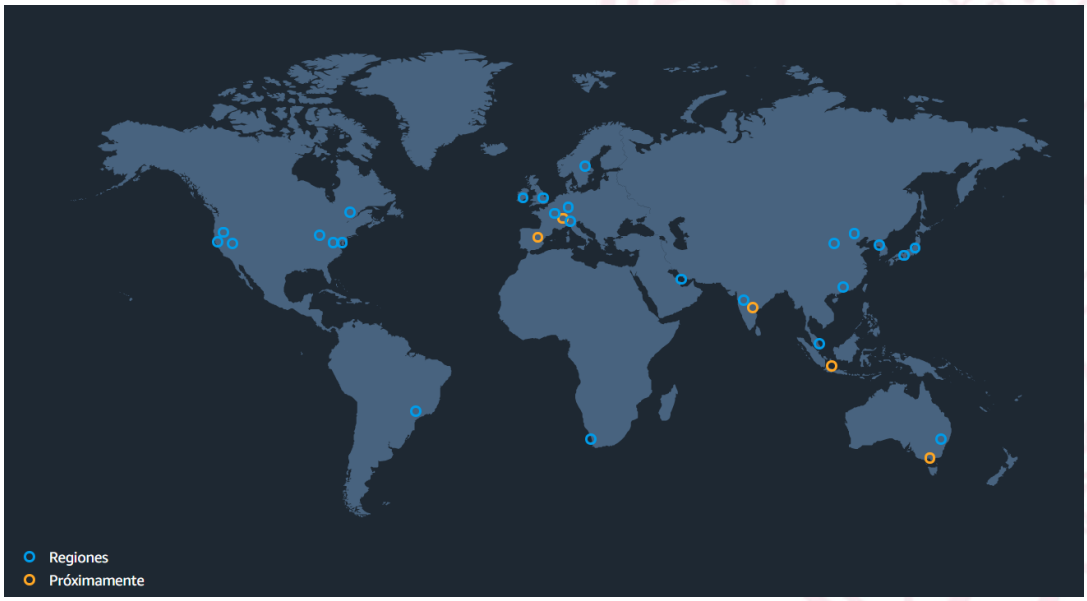

Imagen 13. Regiones en las que se sitúan los centros de datos de Amazon Web (17).

Como dato de interés general hay que destacar la próxima apertura de zonas de disponibilidad dentro de España, puesto que en la actualidad la región más próxima que nos ofrecen Amazon es la de París, Francia.

Google Cloud Platform

Google Cloud es otra de las aplicaciones que se encuentran en la cúspide del sector. Ofrece aproximadamente 90 servicios de tecnología de todo tipo. Al igual que AWS, Google Cloud posee una interfaz que permite el fácil acceso a los usuarios. Los servicios que se ofrecen se adaptan de forma sencilla a las necesidades de cada usuario (19).

Qué tipo de soluciones nos ofrece Google, en este caso estaríamos hablando de tres: IaaS, PaaS y SaaS. Como ya hemos visto anteriormente cada una de ellas va aumentando los recursos que se ofrecen a los usuarios. Las principales categorías de servicios que nos ofrece Google Cloud son los siguientes (19):

1. Cálculo
2. Redes
3. Almacenamiento y bases de datos
4. Inteligencia Artificial y Machine Learning
5. Big Data
6. Identidad y seguridad
7. Aplicaciones de gestión

Microsoft Azure

Microsoft Azure es una plataforma que ofrece más de 200 productos y servicios en la nube con la idea de ofrecer soluciones a problemas del presente y de futuro. Los servicios que ofrecen son muy similares a las de las empresas que ya hemos comentado, pero se destacan por ser 5 veces más barato que AWS. Se puede pensar que al ser una compañía de Microsoft solo ofrece servicios para Windows, pero no es así, también son compatibles con las tecnologías de código abierto. Evitando así cerrarse a un gran nicho de mercado que hace uso de tecnologías de código abierto (20).

Aúnan muchos esfuerzos en la seguridad de los datos de sus clientes, invirtiendo 1 billón de dólares al año para mantener la privacidad de ellos y protegerlos frente a los ciberataques. Una característica destacable de Microsoft Azure es que los clientes tienen la posibilidad de hacer copias de seguridad automáticas de los datos, para de esta forma mantener a salvo los datos de los clientes en caso de pérdida o sustracción, volviendo a hacer hincapié en la importancia de la seguridad de los datos. Como hemos visto los proveedores de servicios nos ofrecen servicios similares y los precios en muchos de ellos se asemejan bastante. Por ello en la **tabla 2** vamos a recoger tres aspectos importantes a la hora determinar cuál es la que mejor se adapta a nuestras necesidades (20).

Tabla 2. Clasificación de proveedores de cómputo en la nube (21).

	Tipo de Cliente	Coste	Servicios
Amazon Web Services	Cualquiera	Gartner recomienda el uso de una herramienta de terceros para estimar el precio	Ofrece muchas opciones, pero se centra en soluciones de nube pública
Google Cloud Platform	Pequeñas y medianas empresas	Gartner destaca los precios centrados en sus clientes con ofertas y descuentos	Menos servicios, pero con capacidades de IA y ML por delante de la competencia
Microsoft Azure	Empresas	Los descuentos se basan en condiciones particulares	Infraestructura robusta haciendo uso de la nube híbrida

2.2.4. Microordenadores

La Raspberry Pi es la placa más popular que se encuentra en el mercado, pero no es la única y sus competidores son muchos. La primera Raspberry Pi que salió al mercado lo hizo en 2012, fue el modelo Raspberry 1 Model A y se caracterizaba por sus 256 MB de RAM, por tener WiFi y por su coste reducido, 40 €. Los modelos se han ido sucediendo durante estos 9 años hasta la aparición de la Raspberry Pi 4 Model B+, que es la placa más potente hasta el momento que ha desarrollado la marca. Capaz de soportar Gigabit Ethernet, WiFi en doble banda de frecuencia, 4 puertos usb, etc. La popularidad de la Raspberry Pi le ha convertido en una de las placas que más se usa para aprender o desarrollar programas y aplicaciones en entornos educativos. Pero sus competidores no se han quedado atrás y han ido desarrollando sus propias placas muchas de ellas destinada a funciones muy específicas (22).

Una de ellas sería la NVIDIA Jetson Nano, que está orientado a la Inteligencia Artificial y es capaz de proporcionarnos 472 GFLOPs de potencia. Al igual que su potencia su precio también es elevado llegando a costar hasta 150 €. Por otro lado, tenemos placas como Arduino que se centran en la robótica o en las impresoras 3D. Se diferencia de Raspberry en que no tiene procesador, sino que hace la función de un microcontrolador. Finalmente, la placa Rock64 Media Board es una de las que más se asemeja en cuanto a precio a la Raspberry, pero ofreciéndonos un procesador más rápido y el doble de memoria (22).

La aceptación de estas placas se debe a su bajo coste (exceptuando algunos casos), su tamaño reducido, la completa funcionalidad que ofrecen como un ordenador convencional, la adaptación a todo tipo de lenguajes de programación y los variados periféricos que permiten integrar. Pero todo sistema tiene sus inconvenientes, y los microordenadores también tienen los suyos. Hay que tener en cuenta que estas placas tienen sus limitaciones en cuanto a computación, memoria o configuración en contraposición con los ordenadores tradiciones. Una vez seamos conscientes de ello, podemos disfrutar de una tecnología que

permite aprender a desarrollar de una forma sencilla y a un coste bastante más bajo en comparación con los ordenadores convencionales.

2.2.5. Sistemas de gestión de bases de datos

Cuando hablamos de una base de datos lo relacionamos directamente con una información almacenada en un lugar. Pero el término va muchos más allá y hay que distinguir el uso que hace de ello. Efectivamente, una base de datos almacena gran cantidad de información para su posterior procesamiento. Estos sistemas son capaces de gestionar muchos de datos con relativa sencillez, pero es importante que diferenciemos entre la base de datos contenedora de información y el programa que gestiona la BBDD. No todos los programas pueden saber leer este tipo de datos ni los programas que las gestionan se pueden autodenominar como base de datos. Tampoco es cierto que son un grupo de ficheros, puesto que, los archivos de un ordenador no conforman una base de datos (23).

Fue en la década de los años 60 donde el término de base de datos empezó a coger forma, proporcionándonos de esta forma una abstracción entre el *software* y el programa de aplicación. Los primeros modelos consistían en modelos en red y jerarquizados, algo demasiado simple y limitado, esto dio paso a las bases de datos relacionales, un sistema mucho más potente que rápidamente se abrió camino en el mercado. Hoy en día la gran mayoría de sistemas de bases de datos hacen uso de modelos relacionales, pero no todos hacen uso de ese modelo, por lo que describiremos por encima algunos de los tipos de modelos que existen (23):

- **Relacional**: es el caso más común en bases de datos. Hace uso de filas para colocar los datos y permite crear relaciones con las demás filas, presentándolo en bloques o columnas. Por ejemplo: MySQL, MariaDB, Oracle DB, etc.
- **Jerárquica**: los datos se almacenan en forma de árbol, con un relación padre-hijo. Por ejemplo: IMS de IBM.
- **De red**: los datos se enlazan entre sí formando una gran red.
- **Orientada a objetos**: extiende el concepto de modelo relacional añadiéndole el sentido de la herencia (algunos objetos heredan parámetros de otros objetos). Por ejemplo: PostgreSQL (combina los modelos relacional y orientado a objetos)
- **Orientada a documentos**: ofrece la oportunidad de almacenar información en diferentes documentos. Por ejemplo: OrientDB o CouchDB.

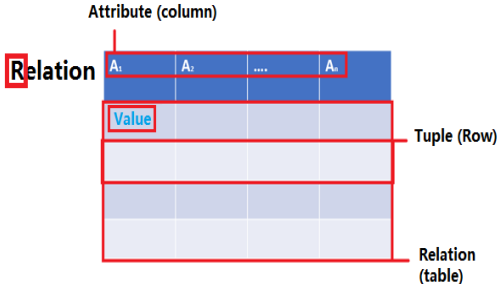

Imagen 14. Modelo de base de datos relacional (23).

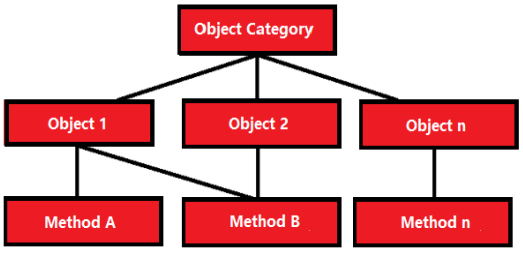

Imagen 14. Modelo de base de datos orientado a objetos (23).

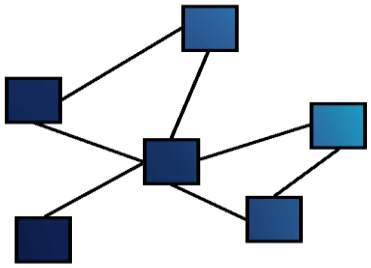

Imagen 16. Modelo de base de datos en red (23).

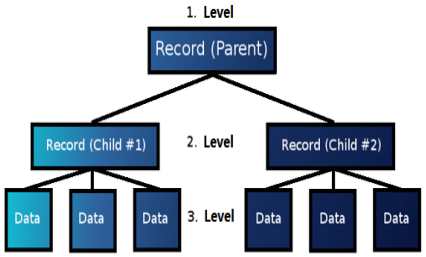

Imagen 15. Modelo de base de datos de forma jerarquizada (23).

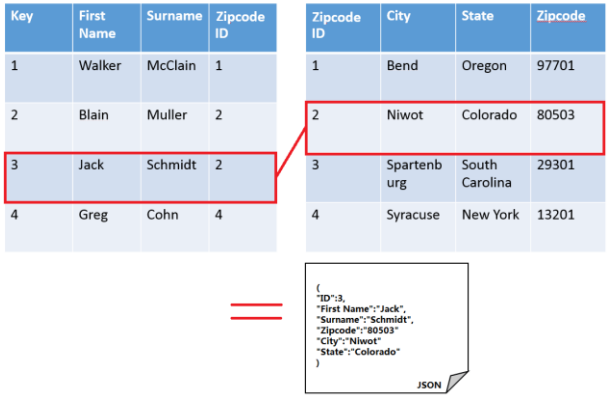

Imagen 18. Modelo de base de datos orientado a documentos (23).

Cuando tenemos los datos en una base de datos debemos de tener en cuenta cómo de protegidos los tenemos y esto cada vez es más importante para las empresas. Toda base de datos debe ser segura y conseguir proteger la disponibilidad, la confidencialidad y la integridad de la base de datos (23):

- **Disponibilidad**: necesidad de mantener los datos siempre a disposición de los clientes.
- **Confidencialidad**: es el aspecto más importante de seguridad dentro de una base de datos y la mejor manera para conseguirlo es tener cifrados los datos.
- **Integridad**: garantizar el acceso a los datos únicamente a las personas autorizadas para ello.

Antes hemos hablado de la importancia de distinguir entre base de datos y el programa que lo gestiona, una vez descritas las características principales de una base de datos, toca describir las funciones principales de un programa de gestión de base de datos y por qué son tan necesarios. Por un lado, el gestor de una base de datos tiene que ser capaz de almacenar, borrar y modificar los datos. Además, tiene que poder gestionar los metadatos que son los que mantienen el orden dentro de la base de datos y contienen la información más relevante. El tratamiento de los datos tiene como base el CRUD (*Create*, *Read*, *Update* y *Delete*). Por otra parte, los datos deben de estar seguros dentro de la base de datos, las consultas deben estar optimizadas, se debe permitir el acceso por varios usuarios simultáneamente y el sistema debe de ser transparente con el usuario (23).

Como hemos comentado antes, existen en el mercado diferentes gestores de bases de datos, por ello vamos a centrarnos en dos de ellas para comentar las características y diferencias que nos podemos encontrar en el mercado.

MariaDB

Aunque MySQL es el gestor de bases de datos por excelencia, el mercado se ha ido expandiendo y han surgido alternativas como MariaDB que estás basadas en *software* libre y con una gran flexibilidad. Es tan cercana a MySQL debido a que su desarrollador fue uno de los que desarrolladores de MySQL, con la idea de mantener las características de MySQL pero en formato de *software* libre. Todo lo bueno de MySQL está integrado en MariaDB pero además incluye mejoras en las consultas, permitiendo crear consultas más complejas y con la opción de mantenerlas en una caché. Permite acceder a clúster de datos lo que provoca que se pueda trabajar muy bien desde la nube. Gracias a la comunidad que se encuentra en continua investigación y desarrollo, MariaDB sigue mejorando y ampliando sus funcionalidades y la seguridad. MariaDB no se ha convertido en una de las mejores herramientas para gestionar las bases de datos por casualidad, sus características lo avalan (24):

- **Velocidad**: la rapidez en sus consultas se debe al uso del **motor aria**, que, en vez de escribir en disco, lo que hace es almacenar las filas en caché. Además, la eliminación de las conversiones en caracteres innecesarios provocó aumentar la velocidad entre un 1-5%.
- **Extensiones que ofrece**: entre muchas de las que posee, podemos destacar la inclusión de un sistema de autenticación, la posibilidad de elegir un motor de almacenamiento de una tabla, uso de columnas virtuales, etc.
- **Errores y *feedback***: en este sentido los errores se van corrigiendo a medida que aparecen y continuamente están implementando mejoras. El *feedback* o las alertas han ido desapareciendo hasta mostrar únicamente las necesarias para solventar una incidencia.
- **Ayuda y documentación**: gracias a la comunidad que da soporte a MariaDB, tenemos a nuestra disposición una completa documentación que nos permite hacer frente a las dudas que nos puedan surgir.
- **GPL**: La licencia GPL es la clave para que MariaDB sea de código abierto. Garantiza a los usuarios la disponibilidad de usar, modificar y compartir de manera libre.

Es de gran importancia saber que todos los programas de MySQL van a ser compatibles con MariaDB. Según el tipo de servidor puede ser interesante estudiar la posibilidad de implementar MariaDB puesto que nos ofrece un gran rendimiento, acceso a funciones específicas y adaptarse rápidamente a soluciones específicas como el uso de aplicaciones en la nube.

PostgreSQL

MariaDB es un tipo de modelo relacional, sin embargo, en el caso de PostgreSQL se combina el modelo relacional y el orientado a objetos. Sigue siendo, al igual que MariaDB, con licencia de código abierto y gracias a los colaboradores que actúan de forma desinteresada se mantiene el proyecto actualizado, esta comunidad se autodenomina PDGD (PostgreSQL Global Development Group). La denominación de este proyecto ha ido evolucionando desde

la década de los 80 en el que se conocía como "Postgres". En 1996, el nombre cambió de forma oficial a "PostgreSQL" y es el que se mantiene hasta la actualidad. Este gestor de base de datos se caracteriza por ser uno de los motores más usados en la actualidad y en parte gracias a las siguientes especificaciones que lo avalan (25):

- **Alta concurrencia**: cada acceso a la base de datos se ve como un acceso único sin importar que otros estén accediendo a la misma tabla de la base de datos. Se denomina como un sistema *Multiversion Concurrency Control* (MVCC), es un método para controlar y proporcionar un acceso concurrente a los datos.
- **Sistema *hot standby***: este proceso permite que se puedan ejecutar búsquedas mientras la base de datos se encuentra en modo *standby*.
- Soporte nativo a diferentes tipos de datos (texto largo, números de precisión, direcciones MAC, arrays, etc.)
- **Uso de formato JSON**: muchas bases de datos encuentran dificultades en los sistemas de bases de datos relacionales. Pero en el caso de PostgreSQL, tenemos un sistema capaz de indexar elementos y realizar búsquedas en dicho formato.
- **Notificaciones**: es un sistema capaz de mantenerse sincronizado con varios dispositivos gracias a las funciones de LISTEN, UNLISTEN Y NOTIFY.
- **Registro**: PostgreSQL hace un registro de cada transacción que se realiza permitiendo restaurar la base de datos desde cualquier punto de restauración.
- **Disparadores**: son procedimientos que se crean con una acción determinada y se ejecutan según unos parámetros.

Al igual que MariaDB, PostgreSQL tiene muchas ventajas. Se pueden clasificar en (25):

1. Uso gratuito
2. Multiplataforma
3. Estable y escalable
4. Incorpora una herramienta gráfica
5. Robusto y fiable
6. Soporte

En contraposición sí que debemos comentar ciertas desventajas de este sistema, por un lado, se destina para bases de datos grandes, en las que se desenvuelve mejor. Finalmente, he de destacar que el lenguaje que emplea presenta dificultades por lo que se recomienda un alto conocimiento del lenguaje SQL (25).

2.2.6. Servidores Web

El paradigma cliente-servidor en aplicaciones web tiene dos partes bien diferenciadas por un lado el cliente y por el otro el servidor. La comunicación entre ambas partes se rige por el protocolo de comunicación HTTP, desde el cliente se envían las peticiones y desde el servidor se procesan las peticiones y se responde en consecuencia (26).

En la **imagen 19** podemos observar un claro ejemplo de funcionamiento de la comunicación entre cliente y servidor. Como vemos ambas partes se encuentran, por lo general, en máquinas separadas, cada una de ellas con su propio *hardware* y *software*. En el caso del cliente tenemos el *front-end,* que es lo que sería el navegador web, en el que el usuario únicamente hace uso de una aplicación sin saber quién lo está gestionando por debajo. En este caso, es el servidor, nuestro *back-end*, el que está gestionando esas peticiones de la aplicación y mostrando al cliente lo que está solicitando en ese momento según la programación que lleva implementada el servidor en cada caso (26).

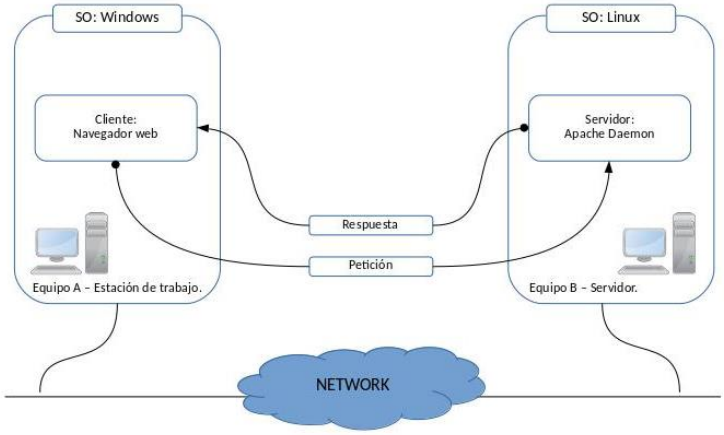

Imagen 16. Ejemplo de estructura básica del paradigma cliente-servidor en aplicaciones web (26).

Al hacer uso de esta tecnología debemos de ser conscientes de las ventajas y desventajas que acarrea este sistema. Primero veamos las ventajas (27):

- **Centralizado**: gracias a tener un sistema central somos capaces de controlar de mejor manera los accesos, los recursos y la integridad de estos. Somos capaces de mantener los recursos actualizados y añadir más de forma sencilla, que por ejemplo en una red P2P.
- **Escalabilidad**: podemos ir creciendo a la vez que lo hacen los clientes adaptando los servidores para que cumplan mejor las exigencias de los usuarios.
- **Mantenimiento**: al tener la posibilidad de crear redundancias entre los servidores, somas capaces también de mantener y aislar de forma sencilla cada uno de ellos en el momento que haya que actualizar o reparar alguno de ellos.
- **Tecnologías**: en el mercado existen muchas soluciones al paradigma cliente-servidor que ofrecen interfaces amigables, fácil implementación y gran cantidad de documentación para llevar a cabo las acciones que el usuario desea realizar.

Por el otro lado hay que ser consciente de las desventajas que pueden tener (27):

- **Gestión del tráfico**: las altas cantidades de tráfico que se pueden dar en los servidores es un problema del paradigma. Cuantos más clientes, más peticiones al servidor y puede que ello cause problemas, en contraposición, existen las rede P2P en el que cada nodo hace el papel de servidor
- **Robustez**: cuando el sistema se cae, las peticiones no pueden ser resueltas. Podemos conseguirlo a través de la redundancia en más de un servidor, que es como funciona las redes P2P, con los recursos distribuidos por sus nodos.

- ***Hardware* y *software***: es muy importante el equipamiento de los servidores y la capacidad de gestionar los recursos. Cuantas más peticiones, más capacidad de gestión por parte de los servidores, lo que incurre directamente en un mayor coste de los dispositivos.

Apache

Uno de los servidores web más populares del mercado es Apache. Se puso en funcionamiento en 1995 y es una herramienta de código abierto, multiplataforma y gratuito. Con él se ejecutan el 46% de aplicaciones web de todo el planeta. Apache no es un dispositivo, sino que es un *software* que se ejecuta en un dispositivo, en un servidor. Su función principal es la de conectar los servidores y navegadores, mientras existe una comunicación entre ellos (28).

Cuando accedemos a una página web lo que estamos haciendo es una petición a un servidor web para que nos muestre la información del sitio web. Apache tiene la función de devolvernos el contenido solicitado, ya sea texto, imágenes, videos, etc. Además, es responsable de que la comunicación sea fluida y segura entre dos máquinas (28).

En cuanto a las funcionalidades de Apache, disponemos de muchos módulos extras que nos permiten, entre muchas otras cosas, configurar la seguridad, almacenar en caché, redirecciones, configurar certificados SSL, crear autenticación, etc. Una vez visto las capacidades de este sistema tenemos que ver los pros y contras que nos ofrece. Por un lado, las ventajas son las siguientes (28):

1. Es código abierto y no supone ningún coste
2. Robusto
3. Mantienen el sistema actualizado
4. Fácil configuración para los principiantes
5. Multiplataforma, disponible en Unix y en Windows
6. Compatibilidad por defecto con WordPress
7. Soporte de ayuda

Por el otro, las desventajas de este servidor web son:

1. Reducción de rendimiento con webs con mucho tráfico
2. Su amplia configuración puede generar brechas de seguridad

NGINX

En la actualidad existen muchos servidores web que están en alza y compiten de tu a tu con Apache, uno de ellos es caso de NGINX. Se lanzó en 2004 y desde entonces ha ido ganando adeptos dentro de los propietarios de aplicaciones web. Fue creado como la solución al problema que se genera en otros servidores cuando el número de solicitudes aumenta hasta las 10.000 simultáneamente (29).

En la **imagen 18**, se puede observar el comportamiento de ambos servidores frente a grandes cantidades de tráfico. Donde constatamos que Nginx atiende a muchas más peticiones que Apache cuando aumenta la cantidad de conexiones. En cambio, Apache se mantiene estable en todo momento (29).

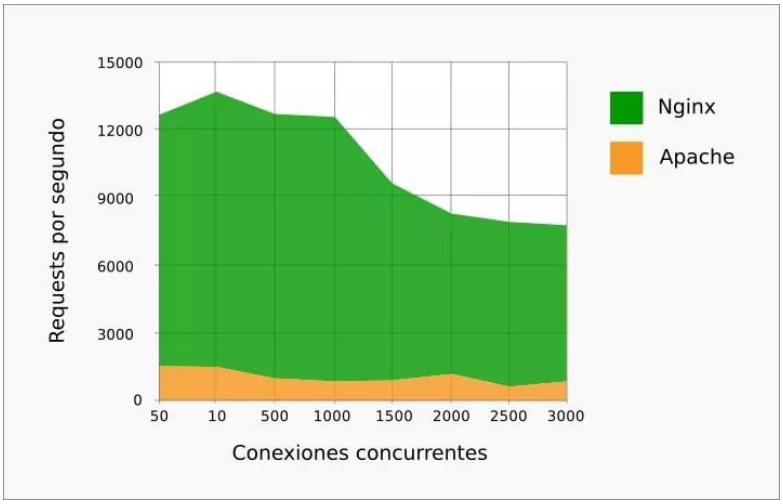

Imagen 17. Comparativa entre Apache y Nginx en cuanto a la capacidad del servidor frente a grandes niveles de concurrencia (29).

Nginx está diseñado para usar muy poca memoria y aceptar altos niveles de tráfico. Su funcionamiento hace que no se genere u proceso por cada petición que se recibe, sino que actúa de forma asíncrona, en donde cada petición es un evento dentro de un solo hilo. Se caracteriza por incluir IPv6, balanceo de carga, websockets, proxy inverso con caché, etc (29).

En la **imagen 21** se puede observar la comparativa de Nginx con los servidores más populares, en ella vemos cuáles son los más utilizados por la comunidad de desarrolladores de aplicaciones web (30).

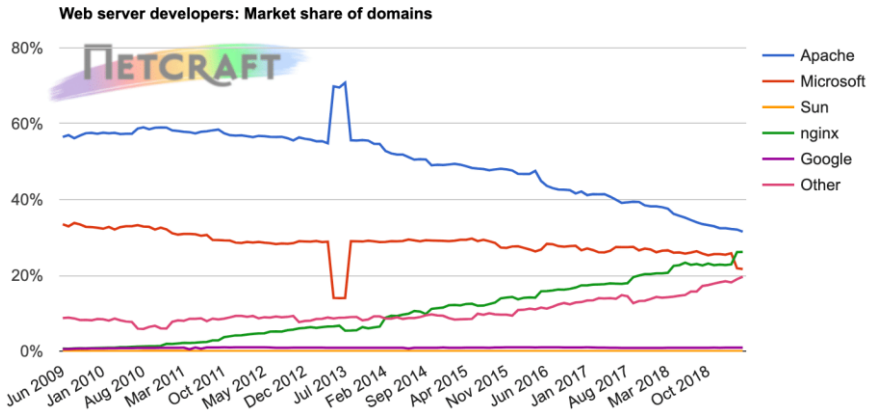

Imagen 18. Comparativa de los servidores web más populares entre 2009 y 2018 (Netcraft) (30).

Apache sigue siendo el más popular de entre todos, pero Nginx sigue expandiéndose por el mercado de las páginas web con mucho tráfico. Un ejemplo de ello es que empresas con Netflix o la NASA hace uso de Nginx en sus servidores (30).

Como se puede observar en las **imágenes 22 y 23**, la popularidad de Apache y Nginx se está igualando. Apache por su parte está teniendo una tendencia a la baja. Sin embargo, Nginx está encontrando la manera de abrirse un hueco (31,32).

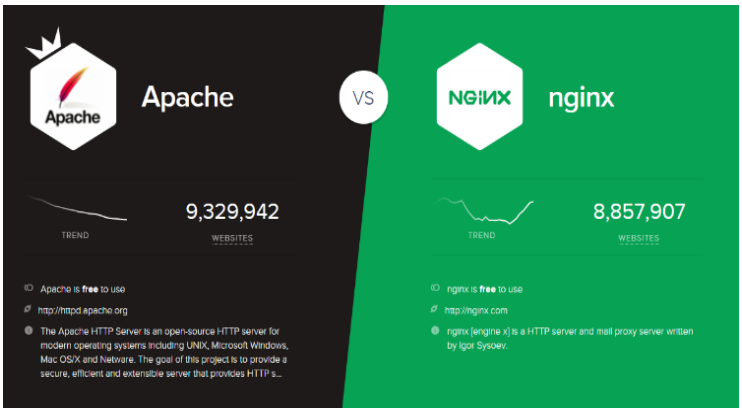

Imagen 22. Comparativa entre Apache y Nginx según el número de páginas web que usan su tecnología en sus servidores (31).

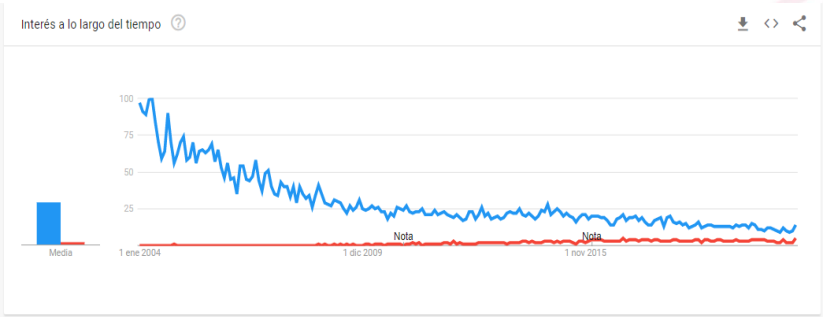

Imagen 23. Comparativa entre Apache y Nginx según su popularidad (32).

3. Requisitos y sistema desarrollado

En este capítulo se van a describir los requisitos tanto funcionales como los no funcionales de este proyecto. Por un lado, nos centraremos en los requisitos funcionales, describiendo las funciones que el sistema debe de tener Y por el otro lado, tenemos los requisitos no funcionales, que están más centrado en el diseño y la implementación.

3.1. Requisitos Funcionales

Como ya he comentado en anteriores capítulos, el sistema que vamos a implementar es un sistema de tele-rehabilitación en el que existen dos partes bien diferenciadas, la parte de los terapeutas y la parte de los pacientes, y que se encuentran unidas por un servidor web alojado en AWS. Una vez descrito el proyecto hay que detallar los requisitos funcionales aportados por las personas que completan el equipo de este proyecto y que a su vez serán usuarios de esta aplicación.

- La aplicación web debe permitir crear, modificar y, en caso de que se tengan los permisos necesarios, borrar registros de la base de datos, ya sea con los formularios de la página web o con el acceso al panel de administración.
- Se deberán incluir roles de acceso al contenido privado de la aplicación. En cada caso, el usuario accederá únicamente a sus pacientes.
- El administrador de la aplicación tiene que ser capaz de administrar los perfiles de los usuarios.
- La aplicación web tiene que ser capaz de filtrar los contenidos para agilizar la búsqueda de los contenidos.
- La aplicación tiene que ser capaz de mostrar el *feedback* de las sesiones de cada uno de los pacientes.
- El acceso a los datos de los pacientes tiene que estar restringido a todo usuario no autorizado.
- Los historiales de los pacientes no pueden ser borrados, se mantendrán como no visibles, lo que supondrá que los terapeutas puedan recuperarlos en un momento determinado.
- Los datos deben mantenerse cifrados y protegidos dentro del servidor.
- El bot mantendrá informados a los padres o tutores de los pacientes para que realicen las sesiones programadas gracias al uso de los recordatorios.
- Se deben realizar copias de la base de datos periódicamente para mantener la integridad de los datos.

3.2. Requisitos No funcionales

Para la evaluación de calidad de un producto *software* tenemos que centrarnos en cumplir con las características que están definidas en la norma ISO/IEC 25010 (33).

- Adecuación Funcional: describe la capacidad de la aplicación en cumplir con las necesidades requeridas en situaciones especificadas.

- Eficiencia de Desempeño: esta característica se refiere a la cantidad de recursos utilizados para llevar a cabo una acción en determinadas ocasiones.
- Compatibilidad: capacidad para que dos sistemas o más puedan intercambiar información o llevar a cabo sus funciones cuando comparten el mismo entorno.
- Usabilidad: capacidad de un producto para ser atractivo, usado y entendido por el usuario.
- Fiabilidad: capacidad de un sistema para ser usado bajo determinadas condiciones y periodos de tiempo determinados.
- Seguridad
- Mantenibilidad
- Portabilidad

3.3. Sistema desarrollado - diseño

En esta sección se describe la estructura *software* del sistema que se ha desarrollado cumpliendo con los requisitos definidos en el capítulo 3. Por otra parte, la arquitectura *hardware* se detalla más adelante en el capítulo siguiente en el que hablamos del despliegue de la aplicación.

3.3.1. Estructura del sistema

El sistema se divide en varios componentes que vamos a describir a continuación:

Aplicación web

Por un lado, tenemos la aplicación web que es la encargada de comunicación entre los terapeutas y los pacientes. Este parte del sistema se ha desarrollado a través de Django. Un framework que nos permite tener estructurada nuestra aplicación web de tal manera que se divide en varias capas: Model-View-Template (MVT). Gracias a ello mantenemos separadas las funciones de la lógica de negocio.

Como podemos ver en la **imagen 24**, Django se estructura en tres partes aisladas entre sí pero unidas a nivel de sistema. A continuación, vamos a describir brevemente la función que cumplen cada una de las partes.

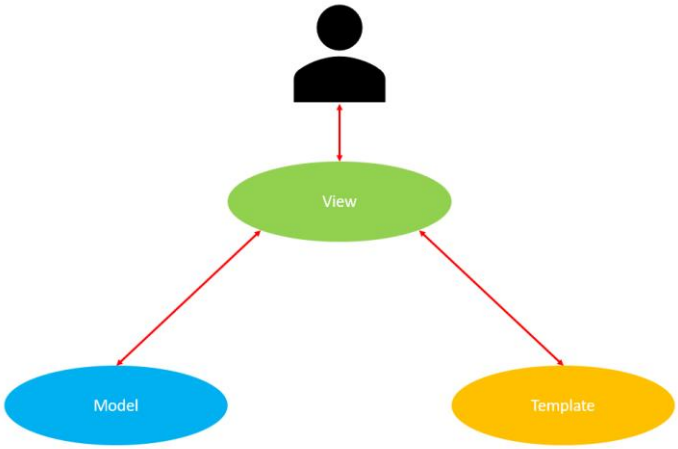

Imagen 19. Arquitectura de Django.

Vista

Es el primer componente con el que de forma transparente nos topamos en la arquitectura del framework. Se encarga de presentar los datos al usuario, es decir, su función es mostrar los datos que el usuario está solicitando. En el caso de nuestra aplicación, es el encargado de mostrar las paginas HTML con el contenido que se solicita en cada caso.

Modelo

Dentro del sistema es el encargado de mantener los datos que van a ser por lo general solicitados durante la visita a la página web. En nuestro caso, se encarga de la base de datos, en concreto, de cada una de las tablas de las que se conforma el sistema y de los datos que hay almacenados en ellas.

Plantilla

En cada una de las plantillas o *templates* se define el estilo y el contenido dinámico que se va a mostrar al usuario en cada una de las vistas. La diferencia con la vista es que el *template* define como se van a ver los datos y la vista define cuales son los datos que se van a ver.

Con esta arquitectura se consigue tener diferenciados los diferentes componentes de la aplicación web consiguiendo tener estructurado todo el contenido. De esta manera, Django facilita mucho el desarrollo de aplicaciones web y aporta funcionalidades extras para el manejo del usuario final.

Sistema Final

A continuación, vamos a describir visualmente el funcionamiento del sistema final en el lado del terapeuta a través de la aplicación web. En la **imagen 25** podemos ver a página principal cuando los terapeutas acceden con sus credenciales en la aplicación web.

Imagen 20. Página principal de aplicación web.

Dentro de ella podemos ver claramente diferenciadas tres secciones. La primera es la de "Pacientes", esta sección nos permite acceder a la zona de pacientes. Solo se muestran los pacientes que pertenecen al terapeuta que se ha loggeado en cada caso, es decir, los pacientes van asociados a un terapeuta y solo ese terapeuta puede acceder a sus datos.

La segunda sección es la de "Ejercicios", en este caso, accedemos a un listado general de ejercicios que se encuentran disponibles y definidos dentro de la aplicación.

Y finalmente tenemos las "Sesiones", que al igual que con los pacientes solo se pueden visualizar las de tus pacientes asociados. En ellas se muestran los datos de las sesiones, las sesiones que se encuentran activas, los detalles de cada sesión, etc.

En la **imagen 26** estaríamos accediendo a lo que es la zona de los pacientes en la que nos encontramos con un breve resumen de los pacientes. Dentro de ella podemos añadir nuevos pacientes, editar los datos de los pacientes, ver los pacientes que se encuentran ocultos en este momento y además podemos filtrarlos en caso de que sea necesario para optimizar el tiempo de búsqueda de uno de los pacientes.

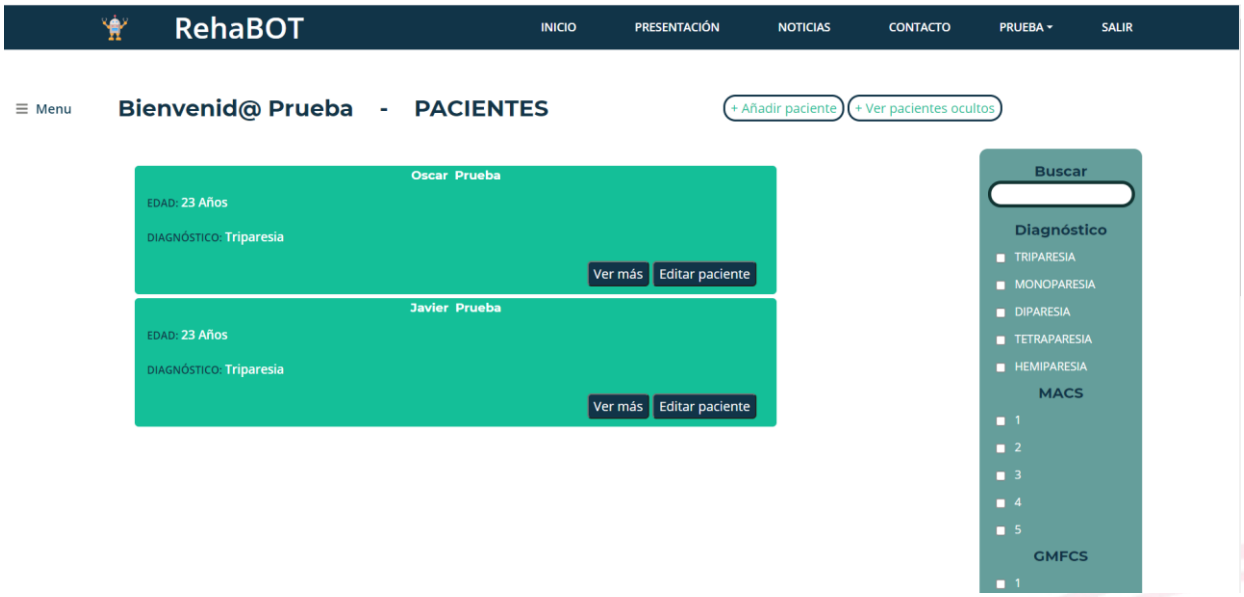

Imagen 21. Página de pacientes de la aplicación web.

Además, en cada una de las fichas de los pacientes podemos ver más información detallada sobre cada uno de los pacientes si hacemos uso del botón "Ver más". Este enlace nos lleva a la **imagen 27** en la que podemos ver un resumen más detallado del historial de nuestro paciente. Podemos ver las sesiones que tiene asociadas, su historial clínico y los datos relacionados con el uso del bot.

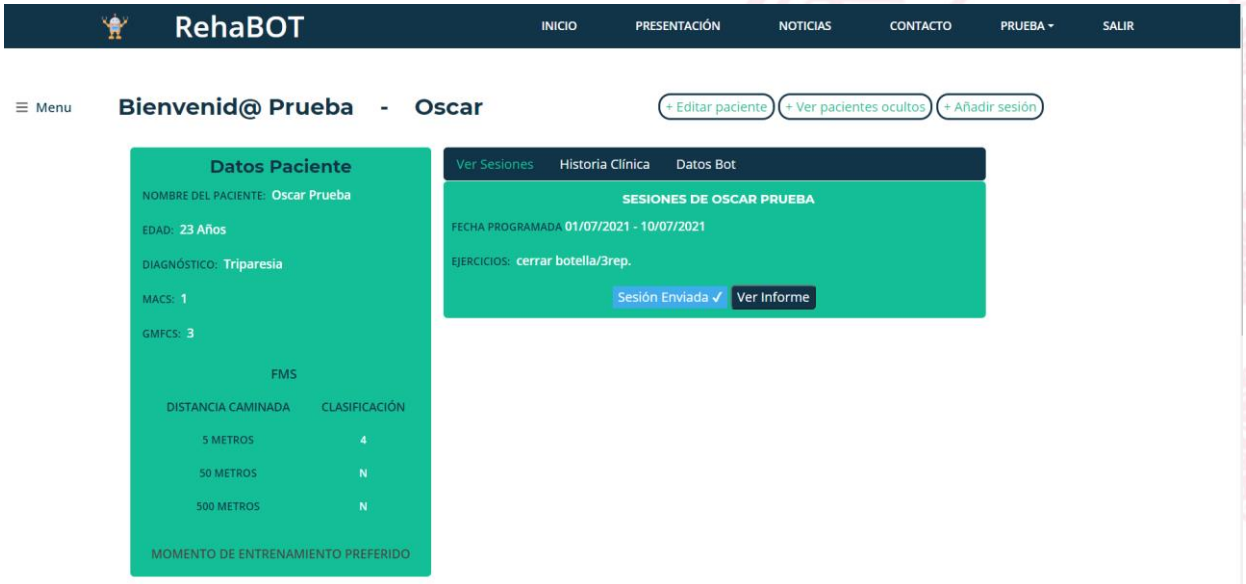

Imagen 22. Página de información detallada de los pacientes en la aplicación web.

En cambio, si accedemos a la sección de ejercicios, veremos algo tal y como lo muestra la **imagen 28** en donde podemos ver con más detalle los ejercicios que han sido dados de alta en el sistema. Al igual que en las secciones anteriores podemos añadir, editar y ver los ejercicios ocultos. Y gracias, nuevamente, a la barra de filtros podemos encontrar el ejercicio deseado en un menor tiempo.

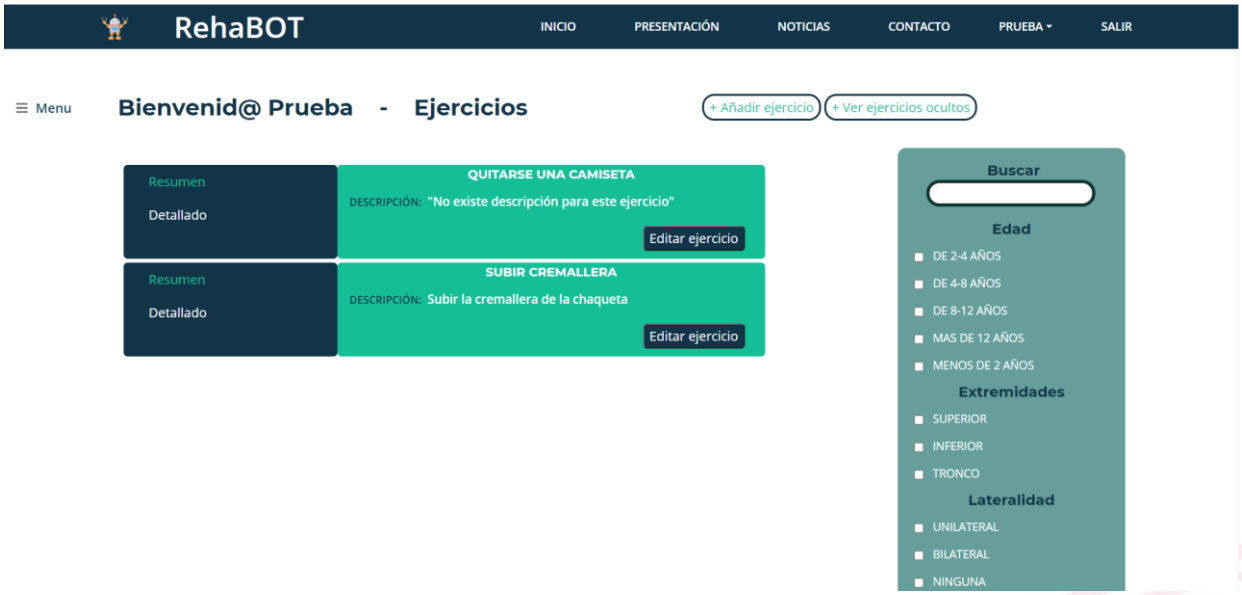

Imagen 23. Página de ejercicios de la aplicación web.

La última de las secciones nos lleva hasta las sesiones de nuestros pacientes donde podemos tener una vista general con las sesiones que se encuentran activas y las que no. En este caso el filtrado es por búsqueda.

Imagen 24. Página de sesiones de la aplicación web.

En las descripciones anteriores hemos estado hablando de ejercicios dados de alta, usuarios asociados a terapeutas, etc. Pero no hemos detallado como el sistema nos permite hacer esto. La forma que tiene Django de hacerlo es a través del panel de administración que podemos ver en la **imagen 30.** Gracias a ello se pueden dar de alta pacientes, ejercicios, sesiones, terapeutas (solo usuarios con permisos para hacerlo). En realidad, el panel de administración permite que se puedan hacer todo tipo de modificaciones, y en este caso muchas más de las que se permiten en la aplicación web pero que por cuestiones de seguridad no estás implementadas.

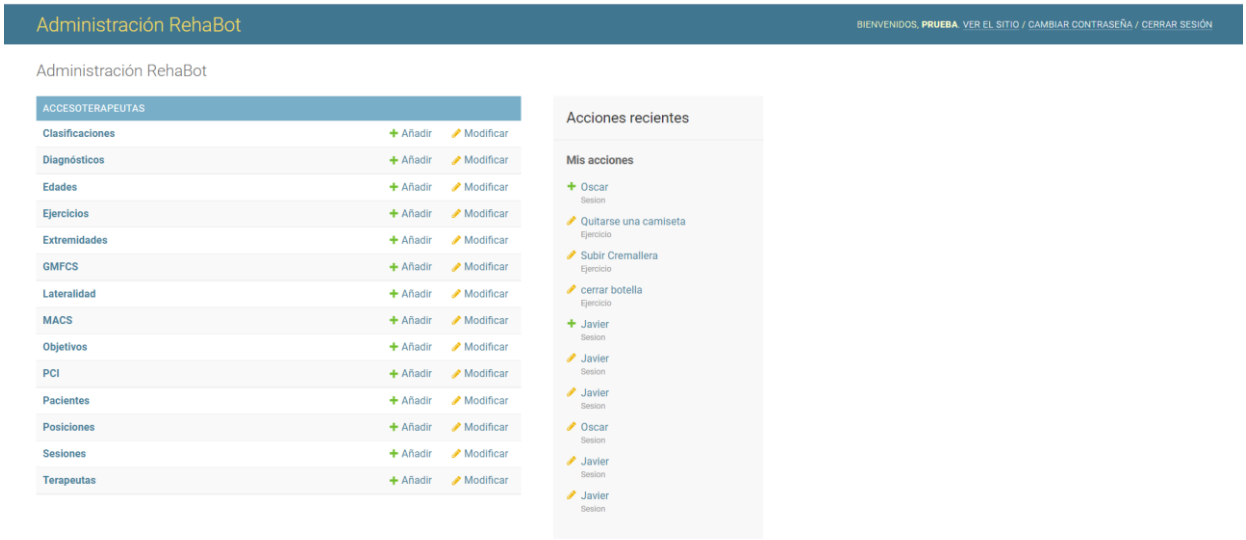

Imagen 25. Panel de administración de la aplicación web.

Base de datos

En este subapartado se va a tratar de explicar cada una de las tablas que da lugar a la base de datos del sistema. Más allá de detallar las relaciones entre tablas vamos a tratar de analizar cada uno de los datos que son almacenados en las tablas y explicar la función que cumplen cada uno de los datos. Gracias a este apartado podremos entender mejor el sistema que se ha diseñado.

Diagnósticos

Esta tabla es la encargada de almacenar los diagnósticos que van a estar disponibles dentro de la aplicación web cuando demos de alta a un paciente o un ejercicio. Indicando por un lado el diagnóstico de un paciente y en el caso de los ejercicios, el diagnóstico al que está sujeto la realización del ejercicio. Un ejemplo de diagnóstico sería: triparesia.

Tabla 3. Diagnósticos

Campo	Tipo de dato	Descripción	¿Es obligatorio?
nombre	*CharField(40)*	Nombre del diagnóstico	Sí
creado	*DateTimeField*	Momento de creación	Auto-rellenado
actualizado	*DateTimeField*	Momento de actualización	Auto-rellenado

Objetivo terapéutico

Los objetivos terapéuticos definen los diferentes grupos de finalidades para las que se lleva a cabo cada uno de los ejercicios, de esta manera podemos definir como objetivo terapéutico el siguiente ejemplo: alimentación. Permitiendo agrupar los ejercicios en grandes grupos destinados a diferentes objetivos definidos en esta tabla.

Tabla 4. Objetivo terapéutico

Campo	Tipo de dato	Descripción	¿Es obligatorio?
nombre	*CharField(40)*	Nombre del objetivo terapéutico	Sí
creado	*DateTimeField*	Momento de creación	Auto-rellenado
actualizado	*DateTimeField*	Momento de actualización	Auto-rellenado

MACS

Esta tabla se corresponde con los diferentes niveles de habilidad manual (de 1 a 5) que se van a definir. La escala MACS permite al terapeuta asociar el nivel de habilidad manual que tiene cada paciente en su historial clínico dentro de la aplicación web.

Tabla 5. MACS

Campo	Tipo de dato	Descripción	¿Es obligatorio?
nombre	*CharField(40)*	Nivel de habilidad manual	Sí
creado	*DateTimeField*	Momento de creación	Auto-rellenado
actualizado	*DateTimeField*	Momento de actualización	Auto-rellenado

GMFCS

La GMFCS, al igual que la MACS, es un valor que el terapeuta rellena al dar de alta al paciente en el sistema. Con él indica la habilidad motora gruesa que posee el paciente en concreto. Lo posibles valores van del 1 al 5.

Tabla 6. GMFCS

Campo	Tipo de dato	Descripción	¿Es obligatorio?
nombre	*CharField(40)*	Nivel de función motora gruesa	Sí
creado	*DateTimeField*	Momento de creación	Auto-rellenado
actualizado	*DateTimeField*	Momento de actualización	Auto-rellenado

Otras calificaciones

Las calificaciones están directamente relacionadas con el campo FMS que se encuentra en la tabla de Pacientes. Su función es definir los niveles de calificación que se dan dentro del Functional Movement Screen (FMS). Los niveles están muy definidos y son los siguientes, del 1 al 6 y las letras C y N.

Tabla 7. Calificaciones

Campo	Tipo de dato	Descripción	¿Es obligatorio?
nombre	*CharField (40)*	Nivel de clasificación de la movilidad funcional	Sí
creado	*DateTimeField*	Momento de creación	Auto-rellenado
actualizado	*DateTimeField*	Momento de actualización	Auto-rellenado

Edad

Esta tabla refleja los rangos de edades disponibles en la aplicación web para dar de alta un nuevo ejercicio. Sirven para agrupar los ejercicios por grupos de edades. Un ejemplo de rango de edad es el siguiente de 2-4 años.

Tabla 8. Edad

Campo	Tipo de dato	Descripción	¿Es obligatorio?
nombre	*CharField (40)*	Rango de edad	Sí
creado	*DateTimeField*	Momento de creación	Auto-rellenado
actualizado	*DateTimeField*	Momento de actualización	Auto-rellenado

Extremidades

Las extremidades son valores que forman parte de los parámetros de un ejercicio. De esta forma indicamos las extremidades que están involucradas en la realización del ejercicio. Como, por ejemplo: superior o inferior

Tabla 9. Extremidades

Campo	Tipo de dato	Descripción	¿Es obligatorio?
nombre	*CharField (40)*	Nombre del tipo de extremidad	Sí
creado	*DateTimeField*	Momento de creación	Auto-rellenado
actualizado	*DateTimeField*	Momento de actualización	Auto-rellenado

Lateralidad

Al igual que en la anterior tabla, la lateralidad forma parte de la parametrización de cada uno de los ejercicios. Los ejercicios van a poder ser unilaterales, bilaterales o ninguna de las dos.

Tabla 10. Lateralidad

Campo	Tipo de dato	Descripción	¿Es obligatorio?
nombre	*CharField (40)*	Nombre del tipo de lateralidad	Sí
creado	*DateTimeField*	Momento de creación	Auto-rellenado
actualizado	*DateTimeField*	Momento de actualización	Auto-rellenado

Posición

Respecto a esta tabla podemos definir en ella la posición del cuerpo que implica la realización del ejercicio. Es un detalle más dentro de la creación de los ejercicios y añade indicaciones para la realización del mismo. Un ejemplo de posición sería “De pie”.

Tabla 11. Posición

Campo	Tipo de dato	Descripción	¿Es obligatorio?
nombre	*CharField(40)*	Nombre del tipo de posición	Sí
creado	*DateTimeField*	Momento de creación	Auto-rellenado
actualizado	*DateTimeField*	Momento de actualización	Auto-rellenado

PCI

PCI se define como el trastorno del tono postural y del movimiento, de carácter, secundario a una agresión no progresiva a un cerebro inmaduro (34). En esta tabla se recoge los tipos de PCI que existen, por ejemplo, espástica, atetoide, atáxica, etc.

Tabla 12. PCI

Campo	Tipo de dato	Descripción	¿Es obligatorio?
nombre	*CharField(40)*	Nombre del PCI	Sí
creado	*DateTimeField*	Momento de creación	Auto-rellenado
actualizado	*DateTimeField*	Momento de actualización	Auto-rellenado

Antes de continuar con las siguientes tablas hay que comentar ciertos aspectos que caracterizan a todas las tablas que acabamos de ver. En primer lugar, todas poseen tres campos en lo que lo único que varía entre ellas es el significado del campo “nombre”.

Puede parecer que son innecesarias algunas de ellas, pero están diseñadas para que puedan crecer a la vez que lo hace el proyecto. Por ello, este diseño se ha hecho pensando en crear una buena base de crecimiento de la aplicación en el que añadir más campos en alguna de las tablas sea sencillo, es decir, buscado un diseño perfectamente escalable.

A partir de la siguiente tabla nos encontraremos con 5 tablas que comienzan a crear relaciones uno a uno y varios a varios con las diferentes tablas que hemos creado con anterioridad. Podemos considerarlas el núcleo de nuestra aplicación y son las siguientes:

Ejercicios

Cada uno de los ejercicios que se dan de alta en la aplicación necesitan antes ser parametrizados con los campos que podemos ver en la tabla posterior. De gran importancia es el campo “código” puesto que es necesario que exista una relación entre el código y el nombre del fichero que se asocia a este ejercicio y se adjunta al campo “video”. Ambos deben coincidir para el correcto funcionamiento de la aplicación.

El resto de los campos, exceptuando el campo “nombre” y *descripción*, se eligen de listas desplegables que facilitan la creación de nuevos ejercicios. Algunos permiten elegir solamente un valor y otros varios valores, es decir, algunos campos son *ForeignKey* y otros son relaciones *ManyToMany.* Todos los campos son obligatorios excepto el campo *descripción* ya que en algunos casos el nombre del ejercicio se puede considerar suficientemente explicativo.

Tabla 13. Ejercicios

Campo	Tipo de dato	Descripción	¿Es obligatorio?
código	*CharField (8)*	Código con el que se identifica el ejercicio	Sí
nombre	*CharField (60)*	Nombre del ejercicio	Sí
descripción	*TextField (500)*	Descripción del ejercicio	No
edad	*CharField (40)*	Selección del rango de edad	Sí
extremidades	*CharField (40)*	Selección de las extremidades	Sí
lateralidad	*CharField (40)*	Selección de la lateralidad	Sí
posición	*CharField (40)*	Selección de la posición	Sí
objetivo terapéutico	*CharField (40)*	Selección del objetivo terapéutico	Sí
diagnostico	*CharField (40)*	Selección del diagnóstico	Sí
pci	*CharField (40)*	Selección del PCI	Sí
video	*FileField*	Subida del video (formato .mp4, .mkv, .avi, etc)	Sí
visible	*BooleanField(Por defecto: True)*	Permite hacer no visible un ejercicio en una situación determinada	Sí
creado	*DateTimeField*	Momento de creación	Auto-rellenado
actualizado	*DateTimeField*	Momento de actualización	Auto-rellenado

Terapeutas

La tabla de terapeutas contiene a todos los terapeutas que poseen credenciales para acceder a la aplicación web. Previamente debe de haberse creado un perfil para cada uno de ellos gracias las facilidades que ofrece Django y a sus tablas en las que almacena a los usuarios y grupos de usuarios. Es una forma de unificar a todos los terapeutas en una tabla ya que en un futuro pueden existir más usuarios que no tienen por qué ser terapeutas y de esta manera lo mantenemos separado.

Tabla 14. Terapeutas

Campo	Tipo de dato	Descripción	¿Es obligatorio?
nombreUsuario	*CharField (30)*	Nombre de usuario del terapeuta para el acceso a la app	Sí
nombre	*CharField (20)*	Nombre del terapeuta	Sí
apellidos	*CharField (40)*	Apellidos del terapeuta	Sí
creado	*DateTimeField*	Momento de creación	Auto-rellenado
actualizado	*DateTimeField*	Momento de actualización	Auto-rellenado

Pacientes

Los pacientes son dados de alta en el sistema rellenando los siguientes campos que aparecen en la tabla. Muchos de ellos han sido definidos con anterioridad en las diferentes tablas lo que permite que rellenar los campos dependa de desplegar listas con posibles valores. Además, nos encontramos con campos que no he visto antes en ninguna otra tabla y que son específicos de los pacientes, que son el usuario y la contraseña con las que se van a identificar frente al bot. Si los datos son erróneos, el control de acceso integrado en el bot nos va a denegar el acceso a nuestra tele-rehabilitación.

Por último, cada paciente está asociado a un terapeuta. Como ya hemos explicado anteriormente, solo el terapeuta que este asociado a un paciente puede ver sus sesiones, su perfil, etc. Por eso es necesario crear esta unión de campos en esta tabla.

Tabla 15. Pacientes

Campo	Tipo de dato	Descripción	¿Es obligatorio?
nombre	*CharField(20)*	Nombre de usuario del terapeuta para el acceso a la app	Sí
apellidos	*CharField(40)*	Apellidos del terapeuta	Sí
fechaNacimiento	*DateField*	Fecha de nacimiento del paciente	Sí
diagnostico	*CharField(40)*	Selección del diagnostico	Sí
macs	*CharField(40)*	Selección de las MACS	Sí
gmfcs	*CharField(40)*	Selección de la GMFCS	Sí
calificacion5	*CharField(40)*	Selección de la calificación FMS para 5 metros	Sí
calificacion50	*CharField(40)*	Selección de la calificación FMS para 50 metros	Sí
calificacion500	*CharField(40)*	Selección de la calificación FMS para 500 metros	Sí
usuario	*CharField(30)*	Usuario para el uso del bot	Sí
contraseña	*CharField(10)*	Contraseña para el uso del bot	Sí
terapeuta	*CharField(30)*	Terapeuta asignado al paciente	Sí
visible	*BooleanField(Por defecto: True)*	Permite hacer no visible un ejercicio en una situación determinada	Sí
creado	*DateTimeField*	Momento de creación	Auto-rellenado
actualizado	*DateTimeField*	Momento de actualización	Auto-rellenado

Sesiones

En la tabla sesiones somos capaces de generar una nueva sesión con ejercicios para un paciente. En ella indicamos el paciente al que va dirigida, para cuando está programado su inicio y su fin y los ejercicios con las respectivas repeticiones que debe de realizar. En esta tabla solo se encuentra el campo ejercicios puesto que está diseñado de tal manera que se ha creado una relación *ManyToMany* en la que se ha introducido el campo de repeticiones

(lo podemos ver en la Tabla 17) Además el terapeuta va a poder generar y enviar sesiones cuando crea conveniente gracias al campo "enviado", con el que podrá ver si la sesión se ha enviado al paciente o aún no. Y aunque no aparezca en la tabla, el sistema muestra un mensaje de Activo si la sesión se encuentra aun dentro de las fechas indicadas en la tabla. La forma de mostrarlo es a través de la interfaz de la aplicación web.

Tabla 16. Sesiones

Campo	Tipo de dato	Descripción	¿Es obligatorio?
paciente	*CharField (20)*	Nombre de paciente al que le asignamos la sesión	Sí
ejercicios	*CharField (60)*	Ejercicios que tendrá la sesión	Sí
periodicidad	*IntegerField*	Número de repeticiones de la sesión	Sí
fechaInicial	*DateField*	Fecha de inicio de la sesión	Sí
fechaFinal	*DateField*	Fecha de finalización de la sesión	Sí
terapeuta	*CharField (30)*	Terapeuta asignado al paciente	Sí
visible	*BooleanField (Por defecto: True)*	Permite hacer no visible una sesion en una situación determinada	Sí
enviado	*BooleanField (Por defecto: False)*	Indica si la sesión ya ha sido enviada al paciente	Sí
creado	*DateTimeField*	Momento de creación	Auto-rellenado
actualizado	*DateTimeField*	Momento de actualización	Auto-rellenado

Sesiones Ejercicios

Como ya hemos comentado, esta tabla se encuentra ligada al campo "ejercicios" de la tabla "sesiones", como consecuencia de la relación *ManyToMany* que existe entre ellas. Se compone de tres campos que relacionan los ejercicios con la sesión y se añade como extra el campo de "repeticiones" que nos permite definir el número de repeticiones que debe hacer el usuario por cada ejercicio.

Tabla 17. Sesiones Ejercicios

Campo	Tipo de dato	Descripción	¿Es obligatorio?
ejercicios	*IntegerField*	Nombre de los ejercicios que se asocian a la sesión	Sí
sesiones	*IntegerField*	Indica la sesión a la que se asocian los ejercicios	Sí
repeticiones	SmallIntegerField	Número de repeticiones de cada ejercicio dentro de la sesión	Sí

Hasta aquí habríamos visto una comunicación terapeuta → paciente, en ese sentido, ya que los datos que aparecen en las tablas los han generado los terapeutas para los pacientes. De los cuales no todos los datos son mostrados/enviados a los pacientes, pero si se usan para crear perfiles de ellos en el sistema y definir tipo de sesiones y ejercicios que van a realizar.

Ahora tenemos que hablar de las ultimas tablas que forman parte de la base de datos en el punto en el que se produce la comunicación desde el paciente al terapeuta. Estamos hablando del envío del *feedback* por parte del paciente. Parte del contenido se autogenera gracias a la programación del bot y por lo tanto muchos datos son recogidos y enviados a los terapeutas de forma transparente al paciente. Un ejemplo de ello son las horas en las que el paciente inicia la sesión, empieza un ejercicio o cuántas veces repite el ejercicio.

El contenido de las tablas se introduce en el sistema de forma automática según la programación horaria que este configurada, y por lo tanto, no necesita intervención por ninguna de las partes.

Ejercicios realizados

Los ejercicios que el paciente realiza durante la sesión se registran en la siguiente tabla. Gracias a ello el terapeuta sabe en todo momento si el usuario ha realizado los ejercicios que ha programado para cada sesión. Los datos que ofrece esta tabla son el ejercicio programado, la fecha completa en la que ha realizado el ejercicio y la sesión a la que pertenece ese ejercicio. Este último campo es muy importante ya que asocia el ejercicio con la sesión a la que pertenece.

Tabla 18. Ejercicios realizados

Campo	Tipo de dato	Descripción	¿Es obligatorio?
ejercicio	*CharField (80)*	Nombre del ejercicio que ha realizado el usuario	Sí
fecha	*CharField (30)*	Fecha y hora en la que se realiza el ejercicio	Sí
sesion	IntegerField	ID de la sesión	Sí

Registro de las sesiones

Las sesiones al igual que los ejercicios llevan asociado un registro, en él los terapeutas pueden ver el tiempo transcurrido desde que empezó la sesión hasta que la finaliza. Además de ello, en esta tabla se recogen los comentarios generales que el paciente puede incluir al final de la sesión, como por ejemplo quejas o problemas que le hayan surgido durante la sesión.

Tabla 19. Registro de las sesiones

Campo	Tipo de dato	Descripción	¿Es obligatorio?
fechaI	*CharField (30)*	Fecha y hora de inicio de la sesión	Sí
fechaF	*CharField (30)*	Fecha y hora de finalización de la sesión	Sí
sesion	IntegerField	ID de la sesión	Sí
comentario	*CharField (100)*	Comentarios sobre la sesión para el terapeuta	Sí

Formulario de pacientes

En esta tabla se recoge un formulario con las preferencias para la realización de las sesiones. Por defecto, se completa la primera vez que el paciente inicia el bot. Más adelante siempre tiene la posibilidad de actualizar los datos. Con ello el terapeuta va a saber el día o días preferidos, las horas en las que se va a poder realizar la sesión y el momento del día que prefiere, mañana o tarde.

Tabla 20. Formulario de pacientes

Campo	Tipo de dato	Descripción	¿Es obligatorio?
dia	*CharField (100)*	Días preferidos por el usuario para hacer la rehabilitación desde casa	Sí
momento	*CharField (30)*	Momento del día para realizar la sesión: mañana o tarde	Sí
horas	*CharField (30)*	Horas preferidas para realizar la sesión	Sí
usuario	*CharField (30)*	Nombre del usuario	Sí
idUser	*IntegerField*	ID del usuario	Sí

Valoración de los pacientes

Las valoraciones individuales de cada ejercicio son recogidas y almacenadas en esta tabla. Esta información es utilizada por el terapeuta para conocer de primera mano las sensaciones del paciente. Se recogen al final de cada ejercicio, exceptuando si el ejercicio ya ha sido valorado en esa misma sesión.

Estos datos permiten visualizar la evolución del paciente en cuanto a realización de ejercicios se refiere, ya que se pueden comparar las valoraciones de los ejercicios en tiempos diferentes. Se puede hacer puesto que las valoraciones están sujetas a un ejercicio y a una sesión en concreto, y a su vez existe un registro de fecha en la que se ha producido la valoración.

Tabla 21. Valoración de los pacientes

Campo	Tipo de dato	Descripción	¿Es obligatorio?
usuario	*CharField (100)*	Paciente que realiza la valoración	Sí
ejercicio	*CharField (30)*	Ejercicio que se valora	Sí
valoracion1	*IntegerField*	Valoración de la primera pregunta	Sí
valoracion2	*IntegerField*	Valoración de la segunda pregunta	Sí
valoracion3	*IntegerField*	Valoración de la tercera pregunta	Sí
valoracion4	*IntegerField*	Valoración de la cuarta pregunta	Sí
valoracion5	*IntegerField*	Valoración de la quinta pregunta	Sí
fecha	*CharField(30)*	Fecha y hora de la valoración	Sí
sesion	*IntegerField*	ID de la sesión	Sí

Una vez se han presentado las tablas que forman la base de datos debemos conocer cómo se relacionan entre sí y para ello disponemos de la **imagen 31** en la que podemos ver la relación que existe entre cada una de ellas, los tipos de datos que almacenan, etc.

Imagen 26. Esquema completo de la base de datos.

API Telegram - Bot

Esta sección recoge el diseño de funcionamiento del bot. Hablaremos del funcionamiento de la conversación y describiremos los pasos que sigue el bot para completar la sesión con el paciente. En primer lugar, antes de describir la conversación que va a tener lugar entre el paciente y el bot, debemos de tener configurado correctamente las credenciales tanto en la aplicación web como en Telegram. Para la aplicación web anteriormente hemos comentado las pautas que se deben seguir, pero para Telegram debemos de seguir los pasos que se nos indican en el Anexo 1. En segundo lugar, hay que destacar que cada bot está asociado a una Raspberry Pi, por lo tanto, evitamos que se generen conversaciones cruzadas entre bots y pacientes. Si partimos de que todo está perfectamente configurado, el funcionamiento se ha diseñado de la siguiente manera.

Paso 1

El paciente debe de enviar el mensaje */start* al bot asignado.

Si es la primera vez que se hablan, el bot como no conoce aún al paciente le va a solicitar su usuario y contraseña, y en el caso de que sean correctos accederemos a la conversación. Por el contrario, si las credenciales no son válidas tendrá 3 intentos más para introducirlas correctamente. (Pasamos al paso 2)**.**

Si no es tu primera vez, pasamos al directamente al paso 3.

Paso 2

Si es la primera vez que usas el bot, te va a explicar el funcionamiento y lo que debes de hacer para realizar correctamente las sesiones. Además, en este caso vas a tener que rellenar el formulario de inicio en el que se recoge la información de horarios que prefieres para la tele-rehabilitación. Después pasaremos al paso 3.

Paso 3

Accederemos al menú principal con las 4 opciones disponibles que existen: comenzar los ejercicios, actualizar mi formulario, finalizar la sesión y ayuda.

En estos tres pasos vemos definido el comportamiento general del bot. Una vez escojamos entre una de las opciones tendremos que seguir un flujo u otro. Hay que tener en cuenta que el diseño de la conversación esta creado de tal manera que se sigan los pasos uno a uno gracia al uso de botones en el teclado. De esta forma, evitamos que se produzcan errores en el flujo de funcionamiento debido a comandos mal introducidos

Veamos ahora el funcionamiento de cada una de las opciones del menú principal.

Comenzar los ejercicios

Esta opción nos permite dar comienzo a los ejercicios, dentro del flujo, aunque no esté reflejado, se encuentran numerosas explicaciones sobre los pasos que deben seguir los pacientes para realizar los ejercicios.

En este caso esta opción está diseñada para que los ejercicios se realicen una vez con las repeticiones correspondientes indicadas por el terapeuta. Por lo que los ejercicios irán desapareciendo a la vez que se van completando.

En el caso de que el video no se haya entendido o que se quiera repetir, el paciente siempre tendrá la opción de hacerlo.

Y por si el paciente no quiere continuar con los ejercicios siempre podrá abandonar la sesión, y evidentemente estos datos serán registrados para que pueda tener conocimiento de ello el terapeuta.

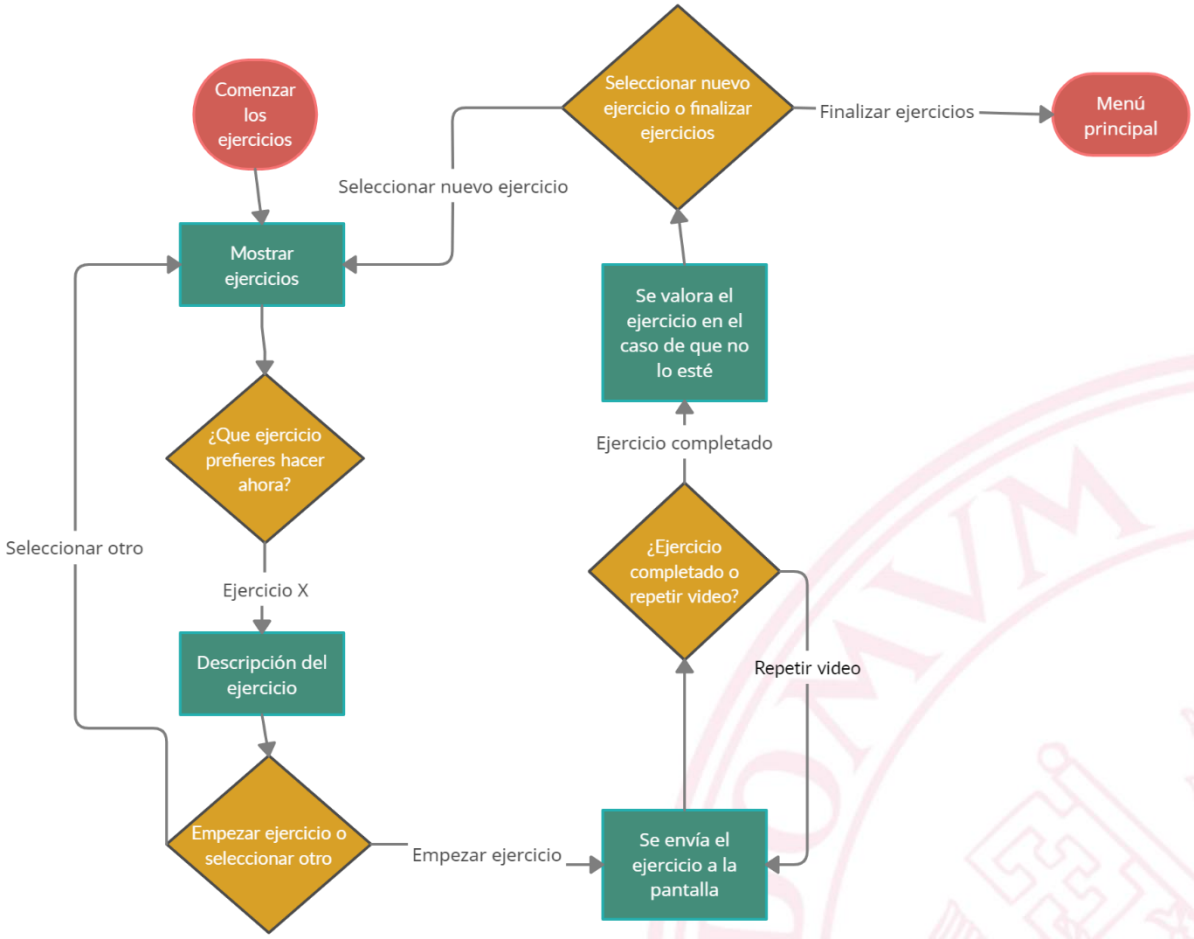

Imagen 27. Diagrama de flujo cuando el paciente comienza los ejercicios.

Actualización del formulario

En el caso de querer actualizar los datos del formulario siempre vamos a tener acceso desde el menú principal para ello. Hay que puntualizar que los dos caminos posibles de la primera pregunta nos llevan a horas dentro del rango de la mañana o en caso contrario a rangos de horas de la tarde según nuestra primera elección.

Tanto la primera como la segunda pregunta son de selección única pero la tercera permite que se seleccione más de un día.

Una vez se han contestado las preguntas se actualiza automáticamente en el servidor, pero no es hasta el día siguiente cuando se hace efectivo, es decir, cuando se actualiza en la base de datos.

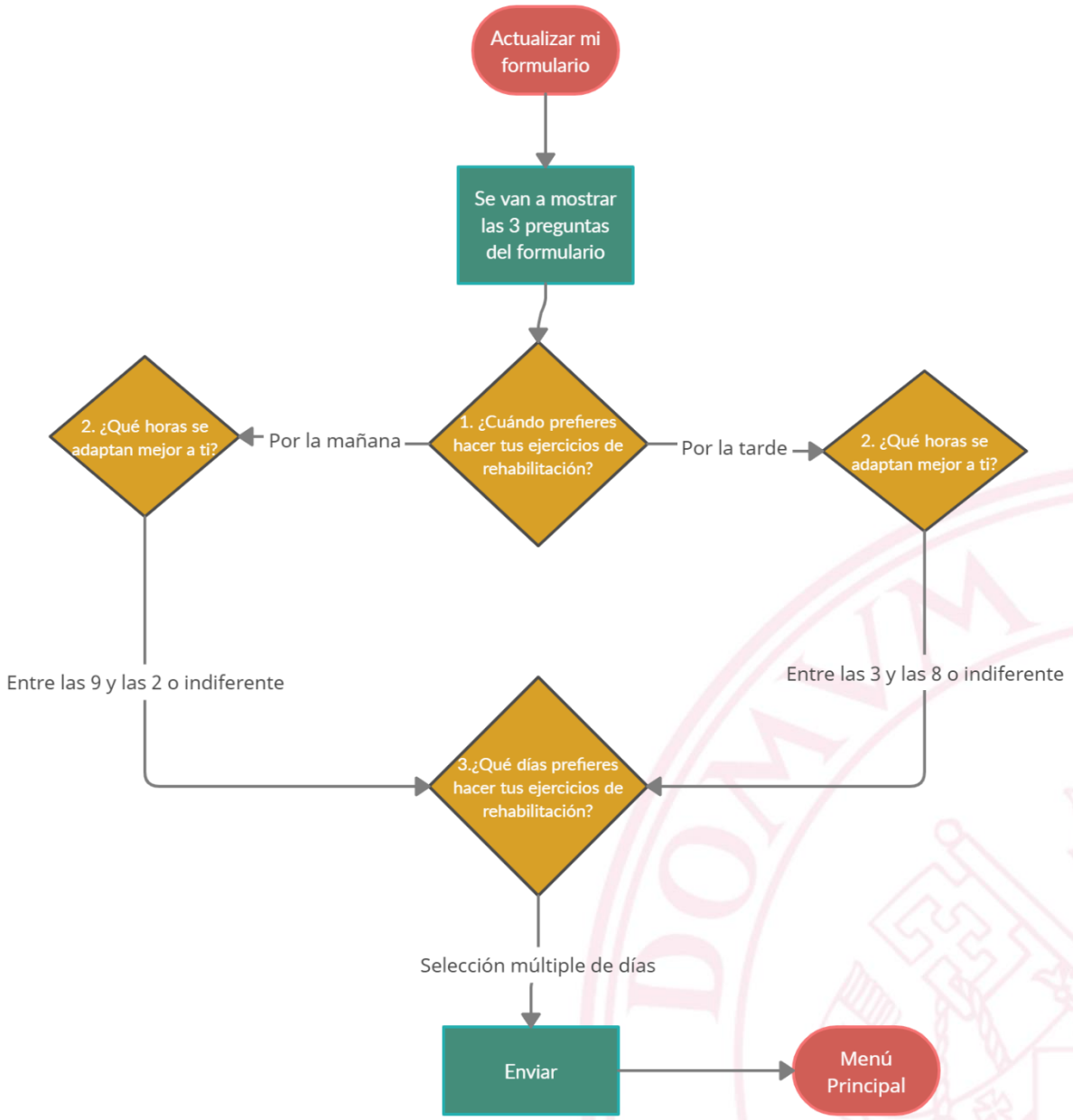

Imagen 28. Diagrama de flujo cuando el paciente actualiza el formulario.

Finalizar la sesión

Esta opción es la que finaliza la sesión por completo y la que permite que se envíen los ficheros al servidor para que después puedan ser introducidos en la base de datos. De esta forma el terapeuta va a poder conocer los datos relativos a la sesión que acaba de realiza su paciente.

Ayuda

Desde ayuda vamos a tener acceso a las instrucciones de uso del bot, en el que se nos describe como hay que manejar el bot y a las indicaciones para cada sesión, en el que se

detalla la información más importante sobre la realización de los ejercicios y las indicaciones del terapeuta.

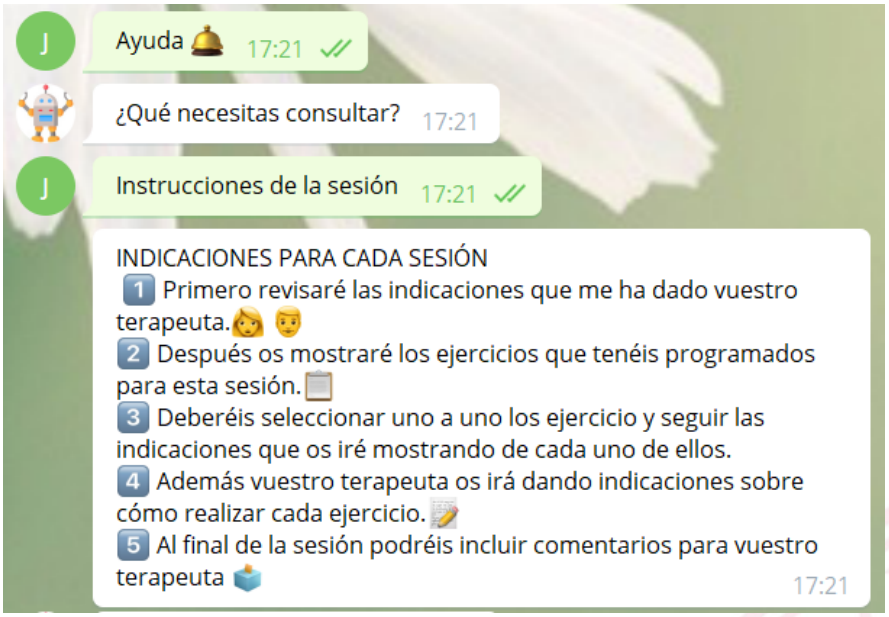

Imagen 29. Mensaje de respuesta del bot cuando el paciente solicita instrucciones de la sesión.

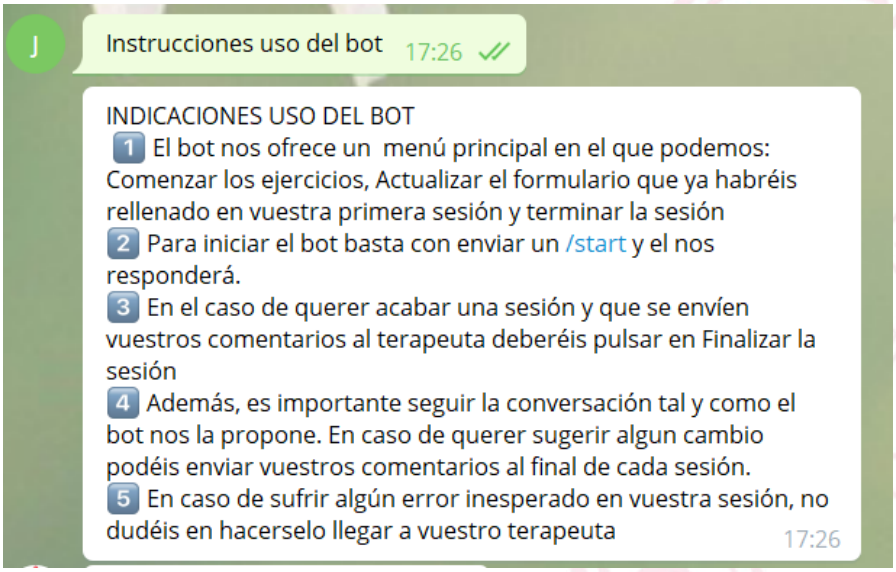

Imagen 30. Mensaje de respuesta del bot cuando el paciente solicita instrucciones de uso del bot.

4. Despliegue

En este capítulo se describe el funcionamiento de la arquitectura de la aplicación, de forma detallada describiremos los componentes de la arquitectura de uno a uno y de forma conjunta ofreciendo una visión global del sistema.

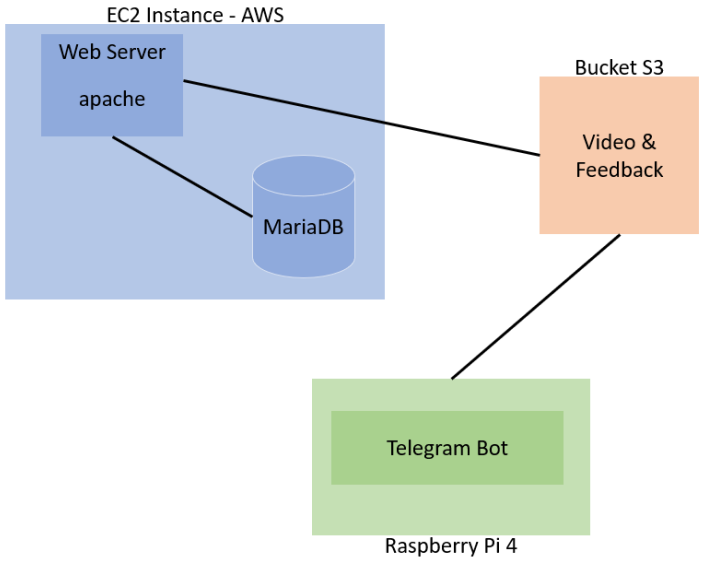

Imagen 31. Arquitectura del sistema desplegado.

El sistema está formado por varios componentes que integran los servicios necesarios para el correcto funcionamiento de la aplicación. Esos componentes los podemos agrupar creando 3 grandes grupos que son los siguientes:

- Amazon EC2
- Bucket S3
- Raspberry Pi 4

En la **imagen 36** podemos ver la arquitectura del sistema con los tres grandes grupos que hemos nombrado anteriormente.

Antes de entrar en detalle, el funcionamiento del sistema se puede resumir de la siguiente manera: cuando un terapeuta desde la aplicación web crea una nueva sesión de ejercicios (se da por sentado que los pacientes están dados de alta en el sistema al igual que los ejercicios, etc.) esta se almacena en la base de datos que esta alojada en la instancia de AWS. Una vez se ha enviado la sesión los datos pasan al Bucket S3 donde se almacena un fichero con los datos correspondientes a la sesión. Dando por hecho que el paciente ya está dado de alta en el sistema y que ya ha configurado los datos en Telegram, basta con que encienda la Raspberry Pi y se ponga en conversación con el bot al que ha sido asignado. De esta manera podrá realizar la sesión que ha sido configurada para él o ella.

Finalmente, los datos que se recogen en los formularios/preguntas durante la realización de la sesión se envían al finalizar la sesión al Bucket S3, en ese momento ya se ha completado el ciclo de comunicación del sistema y el terapeuta recibirá los datos al día siguiente (o cuando se haya programado). Los terapeutas pueden añadir y editar sesiones, pacientes y ejercicios a través de la aplicación web.

Además de todo ello, el sistema incluye sistemas de seguridad para mantener la integridad de la base de datos, gracias en parte al sistema de copias de seguridad y a la monitorización de cambios en los modelos por parte de Django. En las siguientes secciones se desarrollan cada uno de los servicios que acabamos de nombrar y se detallan más a fondo las características de cada uno de ellos.

4.1. Amazon EC2

La instancia

Una vez tenemos diseñado el sistema lo que hay que hacer es crear una instancia. Para ello desde la consola de AWS seguiremos los pasos que nos da el asistente. Primero debemos de seleccionar una imagen para cargar el sistema operativo o lo que en Amazon se denomina una AMI (Amazon Machine Image).

Disponemos de gran cantidad de imágenes, muchas de ellas son específicas para ciertas aplicaciones y otras son más comunes para cualquier aplicación. En nuestro caso hemos seleccionado "Debian 10 Buster".

Después debemos de elegir entre los diversos tipos de instancia que tenemos a nuestra disposición. Las características a tener en cuenta a la hora de elegir un tipo u otro están relacionadas directamente con la funcionalidad que vaya a tener nuestra aplicación. Podemos escoger más memoria menos memoria, más CPU menos CPU… dependiendo de lo que necesitemos. En el proyecto se ha escogido una instancia básica, una *t3.small*, que nos ofrece muy buenas prestaciones para lo que nuestra aplicación va a requerir. Las características son las siguientes:

- 2 CPU virtuales con 2Gib RAM
- 30 GiB de memoria
- Hasta 5 gigabits de velocidad de transferencia de datos

Además, AWS siempre nos ofrece escalabilidad en sus instancias, por lo que cambiar de una instancia a otra no supone ningún esfuerzo si nuestro sistema lo requiere.

Una vez seleccionada la instancia debemos de añadir seguridad a nuestro sistema, esto se hace creando grupos de seguridad dentro de la instancia que actúan como un firewall, creándole reglas. Por lo que disponemos de reglas de entrada y reglas de salida que podemos configurar. En nuestra instancia se han configurado las siguientes reglas:

ENTRADA

- Acceso por el protocolo HTTP desde cualquier dirección IP
- Acceso por el protocolo HTTPS desde cualquier dirección IP
- Acceso por el protocolo SSH desde cualquier dirección IP, para poder acceder a la instancia, configurarla, etc

SALIDA

- Se permite todo el tráfico.

Como acabo de detallar el acceso a la instancia se realiza por SSH, todo ello para poder configurarla, acceder a los servicios y en definitiva controlar a nuestro antojo la instancia.

Cuando acabamos de configurar la instancia es obligatorio obtener un par de claves público-privada para acceder por SSH a nuestra instancia. Y es que es necesario que la comunicación este cifrada para poder acceder a ella.

Para ello se define el Rol IAM (Identity and Access Management) en el que podemos configurar el acceso que damos a cada uno de los usuarios que vayan a acceder a la instancia y los permisos que les damos a cada uno de ellos, gracias a las políticas de acceso. En este caso, soy el único usuario con credenciales y mi rol tiene activada la política de acceso total a la instancia. En cuanto hayamos configurado todo correctamente podemos acceder a nuestra instancia y comprobar que podemos acceder a todos los servicios.

La aplicación

Para acceder a nuestra aplicación web disponemos de un servidor web que se encuentra alojado dentro de nuestra instancia. El servidor web que está funcionando en este sistema es Apache y es el encargado de recibir todas las peticiones del exterior y servir la información a los usuarios.

Para ello lo que debemos de hacer es instalar en nuestra instancia el servidor Apache. Una vez lo tengamos configurado, lo que debemos de hacer es servir los datos de la aplicación web y para ello debemos de conectar Apache con el framework Django. Para realizar esa conexión entre ambas tecnologías lo que debemos de configurar es el archivo *settings.py* de Django y Apache haciendo uso del *mod_wsgi.* Desde la documentación de Django nos ofrecen ya las líneas de configuración que deben de ir ubicadas en la configuración del host virtual dentro de Apache y que son las siguientes:

```
WSGIScriptAlias / /path/to/mysite.com/mysite/wsgi.py
WSGIPythonHome /path/to/venv
WSGIPythonPath /path/to/mysite.com

<Directory /path/to/mysite.com/mysite>
        <Files wsgi.py>
                Require all granted
        </Files>
</Directory>
```

Finalmente, el host virtual configurado para nuestra aplicación web en Apache es el siguiente:

```
<IfModule mod_ssl.c>
 <VirtualHost *:443>

        ServerName www.rehabot.twyncare.com:443
        ServerAdmin infoRehabilitacionBot@gmail.com
        ServerAlias rehabot.twyncare.com

        DocumentRoot /home/admin/rehabilitacion
        <Directory /home/admin/rehabilitacion>
                Require all granted
        </Directory>

        #Cabe destacar que usamos el path a python de nuestro virtualenv
        WSGIDaemonProcess rehabilitacion python-home=/home/admin/developer python-
path=/home/admin/rehabilitacion
```

```
        WSGIProcessGroup rehabilitacion
             WSGIScriptAlias / /home/admin/rehabilitacion/rehabilitacion/wsgi.py

        Alias /static/ /home/admin/rehabilitacion/rehabilitacion_app/static/
        Alias /media/ /home/admin/rehabilitacion/media/

       <Directory /home/admin/rehabilitacion/rehabilitacion_app/static/>
            Require all granted
       </Directory>

       <Directory /home/admin/rehabilitacion/media/>
           Require all granted
       </Directory>

       <Directory /home/admin/rehabilitacion/rehabilitacion>
               <Files wsgi.py>
                       Require all granted
               </Files>
       </Directory>

      ErrorLog ${APACHE_LOG_DIR}/error.log
      CustomLog ${APACHE_LOG_DIR}/access.log combined

      Include /etc/letsencrypt/options-ssl-apache.conf
      ServerAlias www.rehabot.twyncare.com
      SSLCertificateFile /etc/letsencrypt/live/rehabot.twyncare.com/fullchain.pem
      SSLCertificateKeyFile /etc/letsencrypt/live/rehabot.twyncare.com/privkey.pem
</VirtualHost>
</IfModule>
```

Además de la configuración de Apache, para que la comunicación entre Django y Apache se pueda realizar debemos de indicar dentro de Django en el fichero *wsgi.py* las siguientes líneas de código que indican los path donde debe buscar Apache los contenidos que la aplicación debe servir:

```
import os
import sys
from django.core.wsgi import get_wsgi_application

# path a donde está el manage.py de nuestro proyecto Django
sys.path.append('/home/admin/rehabilitacion/')

# referencia (en python) desde el path anterior al fichero settings.py
# Importante hacerlo asi, si hay varias instancias coriendo (en lugar de
setdefault)
os.environ['DJANGO_SETTINGS_MODULE'] = "rehabilitacion.settings"

# prevenimos UnicodeEncodeError
os.environ.setdefault("LANG", "en_US.UTF-8")
os.environ.setdefault("LC_ALL", "en_US.UTF-8")

application = get_wsgi_application()
```

Con estos cambios no estaría todo resuelto, sino que Django nos obliga a realizar ciertas variaciones de seguridad en el fichero *settings.py* las cuales se corresponde con:

- *Debug* a FALSE
- Conexiones SSL activadas
- Indicación de los hosts desde los que se tiene acceso
- Parámetros relacionados con las Cookies
- Indicar la URL de nuestra aplicación

Una vez tengamos todo ello configurado, podemos poner en funcionamiento el servidor y deberíamos ser capaces de acceder a ella desde cualquier dispositivo.

La Base de Datos

Respecto a la base de datos de la aplicación también se encuentra disponible dentro de la propia instancia. Como expuse en capítulos anteriores el número de sistemas de gestión de bases de datos es inmenso. En el caso de este sistema se usa MariaDB, ya que nos ofrece una gestión sencilla, consultas muy agiles y seguridad.

Para ello lo primero que debemos de hacer es instalar MariaDB en nuestra instancia. Una vez lo tengamos instalado, debemos de proceder a su configuración. Antes de configurar el acceso debemos de haber creado ya nuestra base de datos como en el siguiente ejemplo:

```
MariaDB [(none)]> CREATE DATABASE <nombre_base_datos>;
```

Después creamos un usuario y contraseña para acceder a todas las funciones de la base de datos, para ello se usa el siguiente comando:

```
GRANT ALL ON <nombre_base_datos>.* TO 'usuario'@'localhost' IDENTIFIED
BY 'contraseña' WITH GRANT OPTION;
```

Con el anterior comando conseguimos el acceso total a la base de datos que hemos indicado y para acceder a ella necesitaremos usar las credenciales que se han configurado en el comando. Finalmente, para entrar usaremos:

```
mysql -u <nombre_usuario> -p
```

Y el sistema nos solicitará la contraseña.

Una vez tengamos configurado un perfil de acceso a la base de datos debemos integrar en Django las credenciales de acceso. Antes de continuar, quiero destacar que en este sistema vamos a tener 2 gestores de la base de datos. Por un lado, tenemos el sistema nativo de MariaDB, que nos permite realizar cualquier acción sobre la base de datos que nosotros necesitemos, teniendo en cuenta que se trabaja con comandos. Pero por otro lado vamos a tener un acceso a alto nivel gracias a Django. Lo que hacemos es integrar la base de datos (credenciales) en Django y gracias a la arquitectura de Django automáticamente cuando creemos un modelo (tabla dentro de la base de datos) lo que en realidad estamos haciendo es evitar usar un comando para crear esa tabla y usar las opciones que Django incluye para crear tablas y las respetivas relaciones entre ellas.

Todo ello se consigue a través de líneas de código Python muy sencillas que después se integran en un sitio de administración de Django en el que consultar los datos de una de las tablas se consigue haciendo un 'clic'.

Una vez comprendido la estructura de funcionamiento de la base de datos, lo único que hay que hacer es en el archivo de configuración de Django introducir los datos de acceso a la

base de datos que queramos usar. Los datos que se solicitan por parte de Django son los siguientes:

- ENGINE, hay que indicar el tipo de gestor de base de datos que usamos (PostgreSQL por defecto).
- NAME, nombre la base de datos a la que nos vamos a conectar.
- USER, usuario de acceso previamente configurado.
- PASSWORD, contraseña de acceso previamente configurado.
- HOST, por lo general desde 'localhost'.
- PORT, puerto que usemos para acceder a ella.

Después, gracias a los modelos que usa Django, podemos crear las tablas que la aplicación requiera de forma sencilla.

Como medida de seguridad ante el tratamiento de datos clínicos que son de gran sensibilidad, se han programado copias de seguridad de la base de datos en un almacenamiento de AWS, dentro de un *bucket S3*, del cual hablaremos en el siguiente apartado.

4.2. Bucket S3

En los anteriores capítulos hemos hablado de usar *bucket S3* para el almacenamiento de los videos, los ficheros con los datos de la aplicación y las copias de seguridad de la base de datos. De esta forma liberamos al servidor web de tener que almacenar esos datos con lo que ello conlleva.

Lo primero que debemos hacer es acceder a la consola de AWS y entrar al apartado de S3. Una vez dentro seleccionamos la opción de crear un *bucket S3*, nos solicitará un nombre y este debe de cumplir con los siguientes requisitos:

- Ser único en todo Amazon S3
- Tener entre 3 y 63 caracteres
- Sin usar mayúsculas
- Comenzar por una letra minúscula o un número

Después deberemos elegir una región disponible de AWS en la que almacenar el *bucket*. Hay que tener en cuenta que cuanto menor sea la distancia mejor será la experiencia de usuario en cuanto a latencia y coste. En cualquier caso, se podrán transferir de una región a otra según lo necesitemos.

A partir de este momento tendremos la posibilidad de añadir objetos a él. Los tipos de objetos que pueden almacenarse van desde ficheros de texto hasta contenido multimedia. Tendremos también la posibilidad de regular el acceso a los datos gracias al uso de políticas de acceso.

En el siguiente apartado hablaremos de cómo se accede a los datos que están almacenados en el *bucket* de S3 desde la Raspberry Pi, que en realidad se hace de la misma forma que cuando se suben a él desde la aplicación web.

4.3. Raspberry Pi 4

Por último, hablaremos del componente que se sitúa en el lado del paciente, la Raspberry Pi. Este pequeño ordenador es el encargo de interactuar con el paciente y ser su acompañante durante la realización de los ejercicios.

Como sabemos este microordenador se caracteriza por ser muy polivalente a la hora de desarrollar sistemas con cualquier finalidad. Además, su bajo coste lo convierte en una verdadera opción a tener en cuenta cuando queremos desarrollar ciertos proyectos. En este caso, se adapta perfectamente a las necesidades que debe de tener el sistema, pero siempre siendo conocedores de sus limitaciones.

Por otro lado, la elección de esta tecnología se vio condicionada a que nos permite integrar la API de Telegram en el sistema y de esta manera poder desarrollar uno de los puntos clave de nuestro sistema como es el bot de Telegram.

Una vez hemos descrito los requisitos que cumple, veremos cómo es el proceso de despliegue de la aplicación en la Raspberry Pi. Primero, como en cualquier ordenador necesitamos un sistema operativo, en este caso se ha instalado *Raspberry Pi OS with desktop*.

Una vez se configura los parámetros principales del sistema operativo ya podemos acceder a nuestra Raspberry.

Los diferentes paquetes que se han instalado en la Raspberry Pi son los siguientes:

- `git clone --branch master https://github.com/javieru14/BotRehabilitacion`
- `sudo apt install python3 python3-pip python3-setuptools`
- `pip3 install awscli --upgrade --user`
- `Visual Studio Code`
- `pip3 install Schedule`

Para entender mejor cada uno de ellos vamos a ir detallando la función que realizan en nuestra aplicación. El primero de ellos no es más que una clonación de un repositorio en github que contiene la API de Telegram con la que funciona el bot. En el segundo comando, lo que se observa es la instalación de Python que es el lenguaje en el que está desarrollada la aplicación casi por completo, en este caso estamos hablando de versión 3 de Python. Además, se lleva a cabo la instalación de paquetes complementarios para el correcto funcionamiento de Python. En el siguiente comando lo que se instala, como comentamos en el apartado 5.2, la descarga y subida de objetos al *bucket S3* se hace a través de un CLI(*Command Line Interface*) de AWS en el que existen una serie de comandos que nos permiten interactuar con el *bucket*. Los comandos utilizados son los siguientes:

Para la subida:

```
aws s3 sync <source> <target> [--options]
```

Para la descarga:

```
aws s3 sync <target> <source> [--options]
```

Como podemos observar lo único que cambia es el objetivo que tenemos en el comando. Realmente no se suben o se bajan, sino que se define como una sincronización de los objetos

en los que podemos detallar que se sobrescriban o se borren si no existen gracias al apartado de *options*.

En cuanto a la finalidad que le damos a estos comandos es de gran importancia debido a que se encargan de actualizar los ficheros que se "envían" los pacientes y los terapeutas entre ellos. La forma que tienen de ejecutarse es de manera automática por medio de scripts programados a ciertas horas en las que actúan.

Continuando por el cuarto paquete que debemos instalar, nos encontramos con Visual Studio Code. Su instalación no es necesaria, pero es conveniente para hacer comprobaciones de funcionamiento del código del bot en caso de error.

Para finalizar, existe una segunda aplicación que está directamente relacionada con el uso del bot. Se encarga de recordar a los usuarios, en los días preferidos de rehabilitación, que que tienen programada una sesión por parte de su terapeuta. Esta funcionalidad incluye un valor añadido a la aplicación principal ya que mantiene activa la participación de los pacientes en la realización de la tele-rehabilitación. Por ello el paquete *schedule* es tan importante, porque nos permite crear recordatorios basándose en la fecha de finalización de cada una de las sesiones de un paciente y la fecha actual en la que nos encontremos.

Una vez instalados los paquetes que acabamos de describir, el sistema de ficheros de nuestra Raspberry debe de estructurarse según se ha establecido. El sistema está estructurado de tal forma que permite el despliegue en un Raspberry en varios minutos. Para ello se debe de clonar el siguiente repositorio en la raíz del usuario:

```
git clone --branch master https://github.com/javieru14/ScriptsRehabilitacion.git
```

Con ello conseguimos estructurar de manera adecuada todos los scripts que permiten que la aplicación funcione correctamente. Esos scripts son los que sincronizan los datos del paciente y terapeuta a través del CLI de S3 en nuestro *bucket*.

Como sabemos el sistema que se ha desarrollado funciona automáticamente, por ello antes de finalizar el despliegue debemos de programar el funcionamiento de estos scripts que hacen funcionar la aplicación. Hay que tener en cuenta lo critico que es que estos scripts estén en funcionamiento ya que de lo contrario la aplicación podría sufrir fallos. Para ello se han programado una línea de comando por cada script que se debe ejecutar dentro de *crontab*, que no es más que un fichero de texto que contiene tareas programadas. Una vez está todo configurado, el funcionamiento de la aplicación es correcto y el sistema está completo.

4.4. Pruebas de funcionamiento

Una vez tengamos los diferentes componentes del sistema configurados, la aplicación estará lista para usarse. Durante el periodo de pruebas de la aplicación se ha comprobado que la aplicación es robusta pero en ocasiones podremos encontramos con errores o fallos en el funcionamiento. Estos fallos se suelen deber en su gran mayoría a que alguna configuración esta deshabilitada y por ello el flujo de la aplicación da error. Un fallo muy común durante la fase de pruebas ha sido que el sistema no era capaz de reproducir el video, tras depurar el fallo a través de los logs configurados en el bot, el error se debía a una errata en el nombre del fichero que se quería abrir o en cuyo caso una mala sincronización con el servidor. Los fallos similares al que acabamos de nombrar no necesitan una depuración de muchas horas

puesto que si todos los componentes del sistema están funcionando no se va a dar ese problema.

Cuando me refiero a este tipo de fallos, recojo los que han surgido a lo largo de las pruebas y mejoras del bot. Un largo proceso en el que se ha ido perfeccionando el funcionamiento del bot hasta el punto de robustez en el cual se encuentra ahora la aplicación. Por ello es de vital importancia que la configuración se realice de igual forma y en el mismo orden en el que se indica en el capítulo de despliegue.

En cuanto a fallos diferentes a errores de nombres de ficheros o descargas incompletas de ficheros, el usuario puede que se encuentre con situaciones en la que el bot le envié un mensaje indicando: "ERROR: Bot en mantenimiento". En ese caso existen varias formas de solucionar el problema. Teniendo en cuenta que el usuario no tiene por qué saber lo que ha pasado, para recuperar el funcionamiento normal de la aplicación el usuario debe de seguir los siguientes pasos:

1. Apagar la Raspberry Pi
2. Esperar unos segundos hasta que la pantalla este en negro completamente.
3. Encender la Raspberry Pi y esperar unos minutos a que se reinicien todos los servicios necesarios para el funcionamiento del bot.
4. Enviar de nuevo "/start" al bot desde Telegram y comenzar de nuevo la sesión.

Si el error sigue saliendo en el mismo punto, debe comunicárselo a su terapeuta para que se ponga en contacto con el desarrollador, en este caso, conmigo. Gracias al sistema de logs que se ha desarrollado para el bot, sabemos en qué momento se ha producido el error, depurar con mayor facilidad el flujo de la conversación y acometer las acciones necesarias para solucionarlo. La experiencia en la depuración de fallos de la aplicación hace que la gran mayoría de ellos estén relacionados con nombres de ficheros mal escritos, caracteres especiales en los ficheros de las sesiones, videos con nombres repetidos que hacen que el sistema les añada caracteres para diferenciarlos, etc.

Existen otro tipo de fallos que son cometidos por los humanos y que tienen fácil solución. Me refiero a errores, por ejemplo, como los de introducir mal las credenciales de acceso al bot de un paciente. En el caso de que se dé de alta un paciente y este introduzca o reciba unas credenciales que no se correspondan con las suyas, el sistema al tercer intento va a bloquear esa cuenta. Otro fallo similar a este podría ser en el que el usuario no configura adecuadamente su perfil de Telegram, en ese caso, cuando quiera entablar una conversación con el bot, este no le va a responder.

Como podemos comprobar son errores que pueden darse, pero su solución implica poco tiempo y esfuerzo. Para evitar este tipo de fallos, se recomienda que tanto la configuración del perfil de Telegram como una prueba de conversación con el bot realice en la propia visita presencial al terapeuta. De esta forma, podemos evitar que se cometan este tipo de errores y conseguir un uso de la aplicación lo más ideal posible.

5. Conclusiones y líneas futuras

En este capítulo detallaremos las conclusiones que se sacan tras diseñar, desarrollar, implementar y desplegar el sistema. Además, hablaremos del futuro que le depara al sistema en las siguientes actualizaciones de la aplicación.

Conclusiones

Desde el comienzo, el proyecto apuntaba hacia un horizonte con muchas posibilidades en el que necesitábamos desarrollar un sistema desde cero que cumpliera con los requisitos nombrados anteriormente. Tras completar la primera versión del sistema los resultados avalan una buena base para seguir avanzando y trabajando sobre ella. La aplicación se ha desarrollado haciendo uso de tecnologías de bajo coste como la Raspberry Pi y usando como plataforma de implementación la nube de Amazon, AWS; siempre pensando en ser un complemento más para las terapias convencionales. Tras lograr desarrollar esta primera versión del sistema de lo que estamos seguros es de que tiene muchos puntos fuertes que se pueden explotar. La elección de las tecnologías que han sido aplicadas en el desarrollo de la aplicación han sido clave para obtener un prototipo con la rapidez que se ha conseguido. Además, puedo asegurar que la escalabilidad de este proyecto es enorme, lo que le permitirá tener una vida útil más larga para poder seguir siendo desarrollada.

En la actualidad, las aplicaciones que se ejecutan en la nube están a la orden del día y el proceso de digitalización que estamos viviendo permite que podamos estar conectados en cualquier momento a Internet. En el caso de la medicina, los avances son evidentes y nuestro sistema demuestra que es necesario adaptarnos a los nuevos cambios que estamos viviendo. Por ello, es importante que sepamos hacer uso de las herramientas que existen para que la vida de todos sea mucho más fácil.

En este proyecto, lo que buscamos es optimizar los recursos que tenemos. En este caso, basta con tener acceso a internet para que los pacientes puedan rehabilitarse desde casa con el mismo o muy parecido seguimiento que desde las rehabilitaciones presenciales. De esta forma, los pacientes pueden realizar la rehabilitación junto con su terapeuta los días que tengan cita presencial y los días que no tengan cita lo podrán hacer desde su casa, optimizando así los recursos disponibles y progresando a mayor velocidad. De esta manera, el conjunto de tecnologías usadas para crear este sistema permite a los usuarios recibir las sesiones con los ejercicios de rehabilitación para realizarlos desde casa, obtener recordatorios para no olvidarse de las sesiones y enviar comentarios al terapeuta sobre los ejercicios o sesiones programadas. Por otro lado, esta herramienta ayuda a los terapeutas a continuar el seguimiento no presencial de las rehabilitaciones de sus pacientes, pudiendo crear, modificar o consultar la evolución o comentarios de los pacientes.

Este proyecto ha supuesto un reto para mí y me ha servido como crecimiento profesional y personal. He asentado conocimientos en el desarrollo de aplicaciones, he aprendido a encontrar soluciones tecnológicas para las necesidades que acarreaba el sistema y además he sabido indagar en tecnologías que para mí resultaban desconocidas hasta el momento y de las que he adquirido nuevos conocimientos.

Líneas Futuras

Ante un proyecto de este calibre lo que más se espera es conseguir que evolucione a lo largo del tiempo, que incluya mayores servicios, mayor recogida de datos, más funcionalidades

que ayuden a los terapeutas y a los pacientes a continuar con su rehabilitación en cualquier circunstancia y de la mejor manera posible. Por ello desde esta primera versión podemos imaginarnos las líneas por las que el sistema puede crecer:

Pantalla

La pantalla es un accesorio del que se puede sacar mayor rendimiento y mayor usabilidad:

- **Interactividad**: actualmente el uso la pantalla está limitado a los videos. En un futuro deberá convertirse en el accesorio principal que mantenga a los pacientes activos durante las sesiones.
- **Mayores funcionalidades**: es posible que sea interesante convertir la pantalla en una "consola" desde la que funcione el sistema por completo, en la que podamos incluir juegos en los que los ejercicios de la sesión se realicen de igual forma, pero "jugando".
- ***Feedback***: es importante mantener al paciente activo y sería muy buena forma de hacerlo si le vamos ofreciendo *feedback* o puntos al paciente, con la idea de crear recompensas por el esfuerzo que realiza.

En líneas generales aprovecharnos de la interactividad que nos puede ofrecer el uso de cualquier pantalla táctil.

Aplicación Web

En general, el sistema permite una gran escalabilidad y la aplicación web permite abrirse por muchas vías de desarrollo:

- **Visualización**: los terapeutas deben de poder visualizar de forma más sencilla los datos recogidos en la Raspberry Pi, como, por ejemplo, a través de gráficas en el que se recojan las medias de valoraciones de los ejercicios o la evolución de un paciente respecto a la dificultad que encuentra en un ejercicio. De esta forma, los terapeutas son capaces de actuar en consecuencia más rápidamente sobre las necesidades de cada paciente.
- **Funcionalidades**: el sistema va a soportar las necesidades que vayan surgiendo, la implementación de estas será únicamente cuestión de tiempo de programación. Como, por ejemplo, crear un sistema de notificaciones que permita crear alertas según parámetros de satisfacción de sesiones o avisos de finalización de sesiones.

Sensores

Una de las vías próximas de desarrollo se encuentra en la adicción de sensores al sistema. Estos sensores se sitúan en el cuerpo del paciente para recoger los datos de los movimientos durante la realización de las sesiones. El envío de los datos de movimiento recogidos por los sensores mejoraría aun mas la comunicación entre en paciente y el terapeuta, ya que permitiría al terapeuta tener una mejor noción de lo sucedido durante la realización de los ejercicios y la evolución del paciente.

El sistema actual es capaz de evolucionar para conseguir incluir una tecnología más dentro del sistema como son los sensores. Deberán recoger los datos al igual que en la actualidad se recoge el *feedback* de los usuarios, pero además en el servidor deberán de ser analizados

por algoritmos que clasifiquen los movimientos dando así un resultado de la evolución de cada uno de los pacientes a su terapeuta.

Esos datos más adelante pueden ser estudiados para crear modelos de rehabilitación más efectivos adaptándose a cada paciente y sus necesidades. Sin olvidarnos, que el uso de tecnología vestible es flexible y adaptable a otras muchas patologías que requieran de mayor información sobre la evolución de los pacientes.

6. Referencias bibliográficas

1. Dodakian L, McKenzie AL, Le V, See J, Pearson-Fuhrhop K, Burke Quinlan E, et al. A Home-Based Telerehabilitation Program for Patients with Stroke. Neurorehabil Neural Repair. 2017;31(10–11):923-933.
2. Fazzi E, Galli J. New clinical needs and strategies for care in children with neurodisability during COVID-19. Dev Med Child Neurol. 2020;62(7):879-880.
3. Sutter EN, Francis LS, Francis SM, Lench DH, Nemanich ST, Krach LE, Sukal-Moulton T, Gillick BT. Disrupted access to therapies and impact on well-being during the COVID-19 pandemic for children with motor impairment and their caregivers. Am J Phys Med Rehabil. 2021;100(9):821-830.
4. Sadowska M, Sarecka-Hujar B, Kopyta I. Cerebral Palsy: Current Opinions on Definition, Epidemiology, Risk Factors, Classification and Treatment Options. Neuropsychiatr Dis Treat. 2020;16:1505–18.
5. Manifesto for Agile *Software* Development [Internet]. [cited 2021 Jul 29]. Available from: https://agilemanifesto.org/
6. Cikajlo I, Rudolf M, Goljar N, Burger H, Matjačić Z. Telerehabilitation using virtual reality task can improve balance in patients with stroke. Disabil Rehabil. 2012; 34(1):13–8.
7. Kassee C, Hunt C, Holmes MWR LM. Home-based Nintendo Wii training to improve upper-limb function in children ages 7 to 12 with spastic hemiplegic cerebral palsy. J Pediatr Rehabil Med. 2017;10(2):145–54.
8. The Web framework for perfectionists with deadlines | Django [Internet]. [cited 2021 Jul 29]. Available from: https://www.djangoproject.com/
9. Django VS Laravel - Framework Technologies Market Share Comparison [Internet]. [cited 2021 Jul 29]. Available from: https://www.similartech.com/compare/django-vs-laravel
10. 5.2. El patrón de diseño MTV (El libro de Django 1.0) [Internet]. [cited 2021 Jul 29]. Available from: https://uniwebsidad.com/libros/django-1-0/capitulo-5/el-patron-de-diseno-mtv
11. Introducción a Django - Aprende sobre desarrollo web | MDN [Internet]. [cited 2021 Jul 29]. Available from: https://developer.mozilla.org/es/docs/Learn/Server-side/Django/Introduction
12. Telegram Bot API [Internet]. [cited 2021 Jul 29]. Available from: https://core.telegram.org/bots/api
13. Empieza a usar Chatfuel | Chatfuel Documentation [Internet]. [cited 2021 Jul 29]. Available from: https://docs.chatfuel.com/en/articles/5119062-empieza-a-usar-chatfuel
14. Signal >> Página principal [Internet]. [cited 2021 Jul 29]. Available from: https://signal.org/es/#signal
15. Cloud computing: una guía de aproximación para el empresario | INCIBE [Internet]. [cited 2021 Jul 29]. Available from: https://www.incibe.es/protege-tu-empresa/guias/cloud-computing-guia-aproximacion-el-empresario
16. Mell P, Grance T. The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology.
17. ¿Qué es AWS? [Internet]. [cited 2021 Jul 29]. Available from: https://aws.amazon.com/es/what-is-aws/

18. 2020 Magic Quadrant for Cloud Infrastructure & Platform Services [Internet]. [cited 2021 Jul 29]. Available from: https://pages.awscloud.com/GLOBAL-multi-DL-gartner-mq-cips-2020-learn.html?pg=WIAWS
19. Google Cloud Platform: Qué es, cómo se usa y cómo se compara [Internet]. [cited 2021 Jul 29]. Available from: https://www.acronis.com/es-es/articles/google-cloud-platform/
20. Qué es Azure: Servicios en la nube de Microsoft | Microsoft Azure [Internet]. [cited 2021 Jul 29]. Available from: https://azure.microsoft.com/es-es/overview/what-is-azure/
21. Bala AR, Gill B, Smith D, Wright D, Ji K. Licensed for Distribution Magic Quadrant for Cloud Infrastructure and Platform Services. 2020;
22. Placas de computación alternativas a Raspberry Pi 4 [Internet]. [cited 2021 Jul 29]. Available from: https://hardzone.es/reportajes/listas/alternativas-raspberry-pi-4/
23. Bases de datos: historia, funciones y modelos - IONOS [Internet]. [cited 2021 Jul 29]. Available from: https://www.ionos.es/digitalguide/hosting/cuestiones-tecnicas/bases-de-datos/
24. Qué es MariaDB y cuáles son sus características | Blog | Hosting Plus España [Internet]. [cited 2021 Jul 29]. Available from: https://www.hostingplus.com.es/blog/que-es-mariadb-y-cuales-son-sus-caracteristicas/
25. PostgreSQL: ¿Qué es? Características, Ventajas y Desventajas [Internet]. [cited 2021 Jul 29]. Available from: https://hostingpedia.net/postgresql.html
26. Paradigma cliente-servidor en aplicaciones web. — documentación de Vagrant - 0.1.0 [Internet]. [cited 2021 Jul 29]. Available from: https://vagrant-intro.readthedocs.io/es/latest/paradigma.html
27. Ventajas - Desventajas - Cliente Servidor [Internet]. [cited 2021 Jul 29]. Available from: https://robiniclienteservidor.weebly.com/ventajas---desventajas.html
28. ¿Qué es Apache? Descripción completa del servidor web Apache [Internet]. [cited 2021 Jul 29]. Available from: https://www.hostinger.es/tutoriales/que-es-apache/
29. 5 Alternativas a Apache para máximo rendimiento - Infranetworking [Internet]. [cited 2021 Jul 29]. Available from: https://blog.infranetworking.com/alternativas-a-apache/
30. ¿Qué Es NGINX y Cómo Funciona? NGINX explicado para principiantes [Internet]. [cited 2021 Jul 29]. Available from: https://kinsta.com/es/base-de-conocimiento/que-es-nginx/
31. apache, nginx - Explorar - Google Trends [Internet]. [cited 2021 Jul 29]. Available from: https://trends.google.es/trends/explore?geo=ES&q=apache,nginx
32. Apache VS nginx - Web Server Technologies Market Share Comparison [Internet]. [cited 2021 Jul 29]. Available from: https://www.similartech.com/compare/apache-vs-nginx
33. ISO - ISO/IEC 25010:2011 - Systems and *software* engineering — Systems and *software* Quality Requirements and Evaluation (SQuaRE) — System and *software* quality models [Internet]. [cited 2021 Jul 29]. Available from: https://www.iso.org/standard/35733.html
34. Parálisis Cerebral - Fisioterapia Neurológica [Internet]. [cited 2021 Jul 29]. Available from: https://www.fisioterapianeurologica.es/patologias/paralisis-cerebral/

Anexo 1 - Configuración de Telegram para el uso del bot

En este anexo explicaremos los pasos que se deben seguir para completar de forma exitosa la configuración de Telegram.

Usuarios nuevos de Telegram

1. Lo primero que debemos de hacer es descargarnos la aplicación en el teléfono móvil.

Imagen 32. Ejemplo de descarga de Telegram en sistema operativo iOS.

2. Debemos de crearnos una nueva cuenta y para ello nos solicitarán introducir nuestro teléfono móvil y para confirmar que somos nosotros nos enviarán un código de verificación que debemos introducir.

Imagen 33. Configuración de teléfono en Telegram.

Imagen 34. Confirmación del teléfono a través de un SMS.

3. Ya tenemos nuestra cuenta creada.

En este punto ya podemos empezar a configurar nuestra cuenta tanto los usuarios nuevos como los que ya disponían de una cuenta en Telegram. Lo que vamos a configurar es la privacidad de la cuenta y el usuario para poder usar el bot.

Privacidad

1. Ajustes > Privacidad y seguridad > Número de teléfono

De esta forma conseguiremos que nadie pueda ver nuestro número de teléfono. Es recomendable mantener la privacidad de nuestro teléfono, pero no es **OBLIGATORIO** para el correcto funcionamiento del bot.

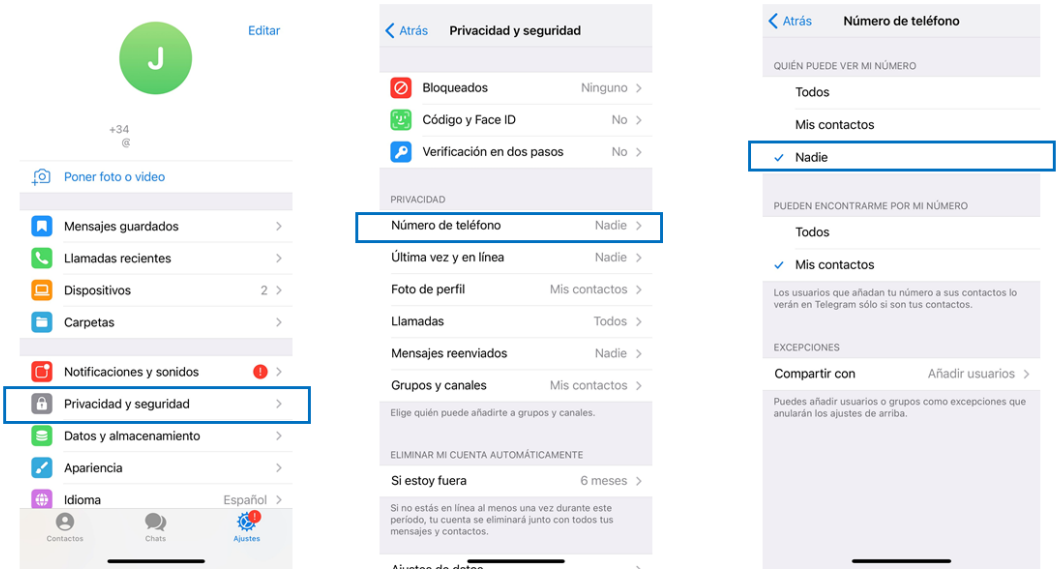

Imagen 35. Configuración de privacidad de número de teléfono en Telegram.

Configuración usuario

Para que el bot permita que accedamos a las sesiones necesita que nuestro nombre de usuario sea igual al que el terapeuta nos ha proporcionado. Servirá tanto para acceder como para después obtener las sesiones que nos programen los terapeutas.

1. Ajustes > Editar > Nombre de usuario > “Rellenamos con el usuario que nos ha proporcionado el terapeuta”.

En el caso que Telegram no nos deje poner ese nombre de usuario deberemos de consultarlo con nuestro terapeuta para que nos ofrezca un nuevo usuario.

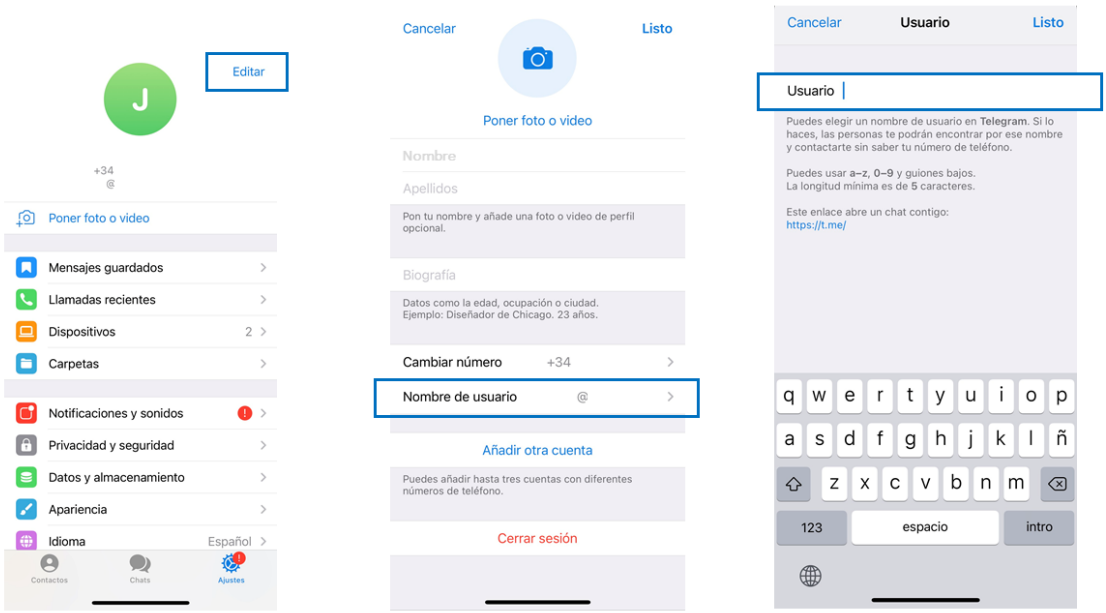

Imagen 36. Configuración de usuario para hacer uso del bot.

Una vez tengamos todo configurado estaremos listos para iniciar la conversación con el bot. Para iniciar una conversación en Telegram debemos de seguir los siguientes pasos.

1. Primero buscamos la persona/bot con la que deseamos entablar una conversación.

Imagen 37. Búsqueda por nombre del bot en iOS.

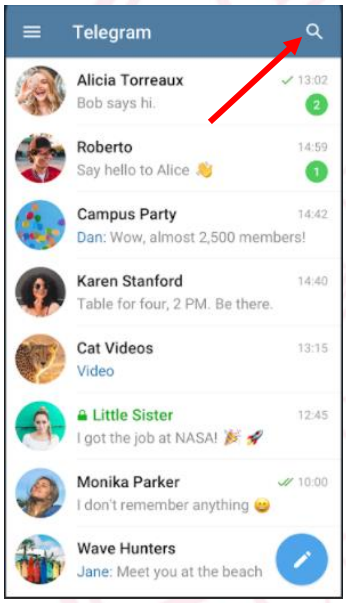

Imagen 38. Búsqueda por nombre del bot en Android.

En ambos sistemas operativos debemos buscar al bot que se nos ha asociado, para ello deberemos de introducir el nombre del bot en el buscador y pichar sobre él.

Los bot tendrán una apariencia similar a la de la **imagen 44.**

Imagen 39. Apariencia del bot con el que se comunican los pacientes.

2. Una vez hayamos encontrado el bot que nos corresponde, lo que debemos hacer para iniciar la conversación es escribir */start* o pulsar en "Iniciar".
3. En nuestra primera sesión nos pedirá la contraseña. Si es correcta, tendremos acceso a nuestras sesiones y el bot nos tendrá recordados para las siguientes sesiones.

Si se produce algún error durante la configuración, por favor, comuníqueselo a su terapeuta.

Anexo 2 – Manual uso de aplicación web

En este manual vamos a explicar que pasos deben seguir los terapeutas para los diferentes casos de uso de la aplicación web:

1. Crear pacientes, ejercicios o sesiones
2. Editar pacientes, ejercicios o sesiones
3. Ocultar pacientes, ejercicios o sesiones
4. Consultar pacientes
5. Consultar sesiones
6. Consultar lista de ejercicios
7. Consultar credenciales para el bot del paciente

Caso de uso 1 - Crear pacientes, ejercicios o sesiones

Para los tres casos la metodología es la misma y en cada uno de ellos tenemos dos caminos posibles para hacerlo.

1. Acceder a la aplicación web > En el caso de querer dar de alta un nuevo usuario pinchamos en el **1**, si lo que queremos es crear un nuevo ejercicio seleccionamos el **2** y si en realidad lo que necesitamos es crear una sesión, pulsamos en **3**.

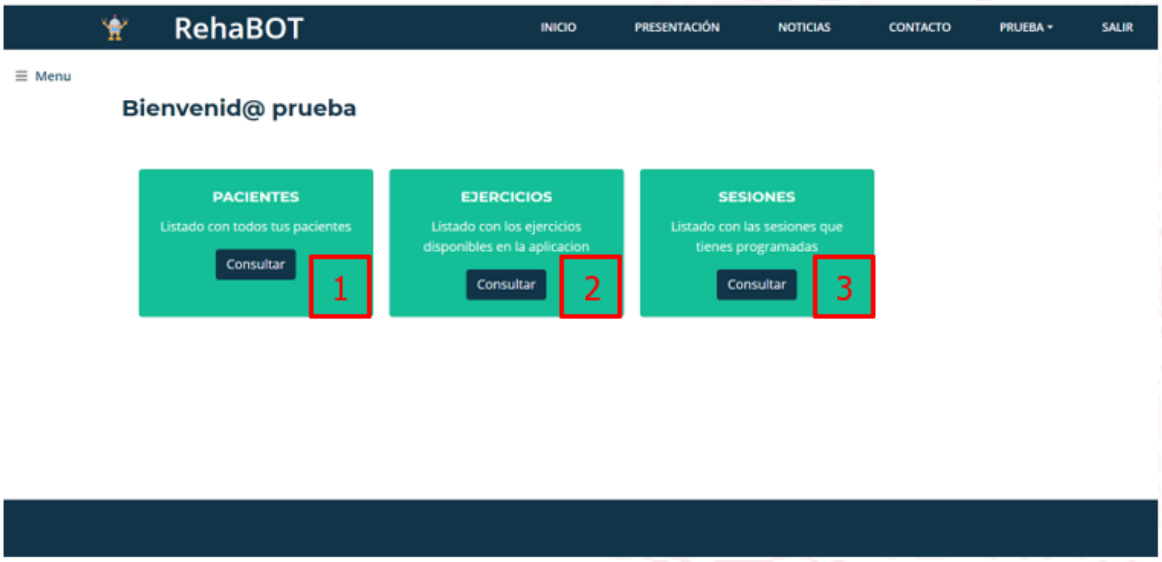

Imagen 40. Página principal de la aplicación web.

2. Dentro de "Pacientes", tendremos la opción en la esquina superior derecha de dar de alta a un paciente nuevo.

Imagen 41. Botón que nos permite añadir un nuevo paciente.

3. Este enlace nos redirigirá al panel de administración de la aplicación al apartado de dar de alta a nuevos pacientes. Solo deberemos de rellenar todos los campos que sean necesarios y para finalizar pulsaremos en "Guardar".

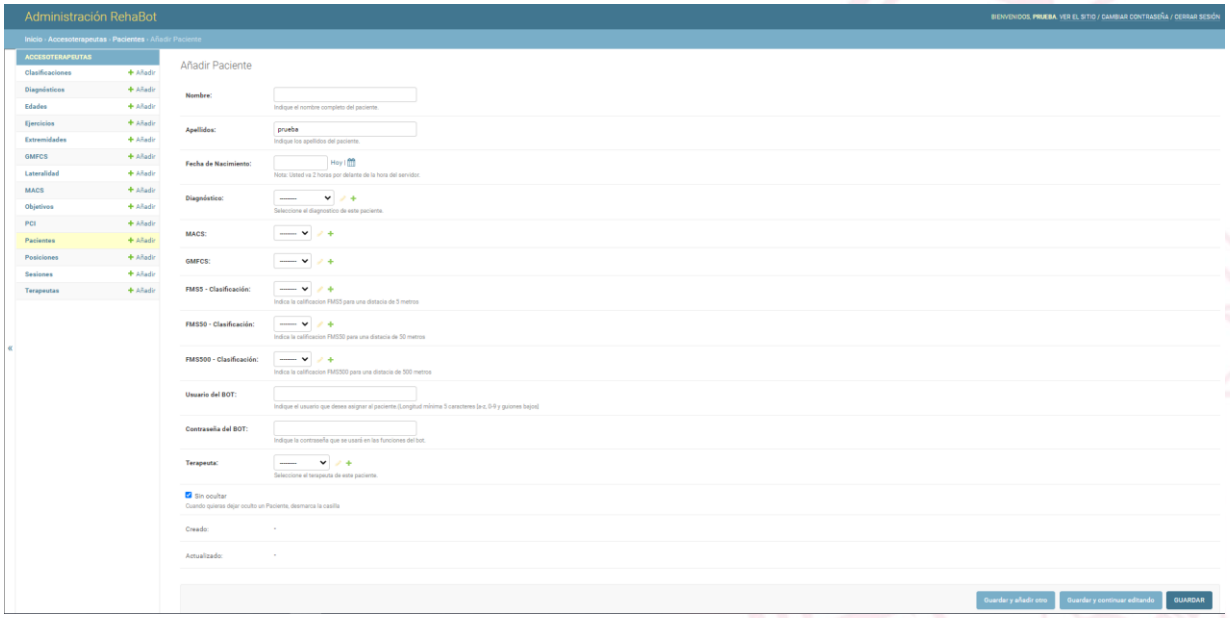

Imagen 42. Panel de administración en el que damos de alta al paciente.

Estos mismos pasos tendremos que hacer si queremos crear ejercicios o sesiones. De tal manera que todo ello lo podríamos hacer directamente accediendo al panel de administración de la aplicación.

Para acceder al panel de administración se puede hacer a través de la siguiente url https://www.rehabot.twyncare.com/admin o también desde el menú principal de la aplicación.

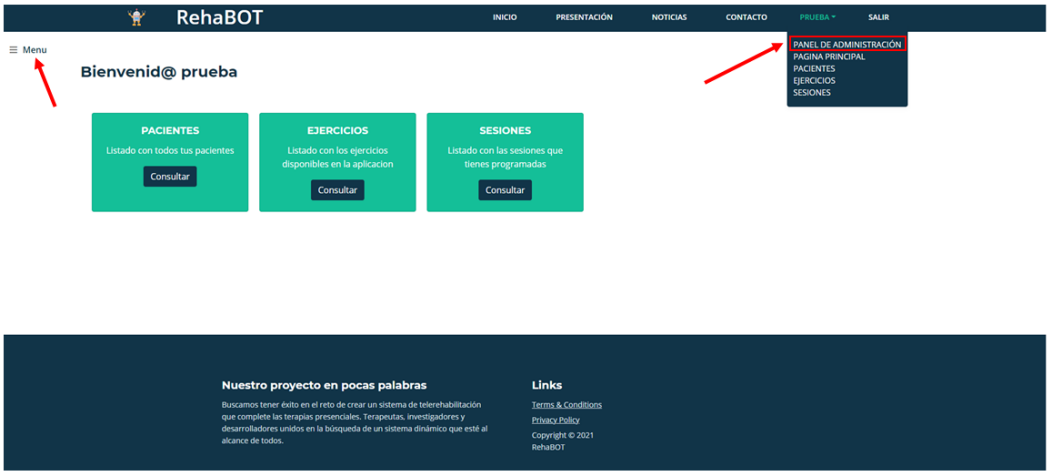

Imagen 43. Dos formas de acceder al panel de administración.

En el caso de no estar *loggeados* en la aplicación nos aparecerá en pantalla la siguiente ventana. Deberemos introducir las mismas credenciales que para acceder desde la aplicación web.

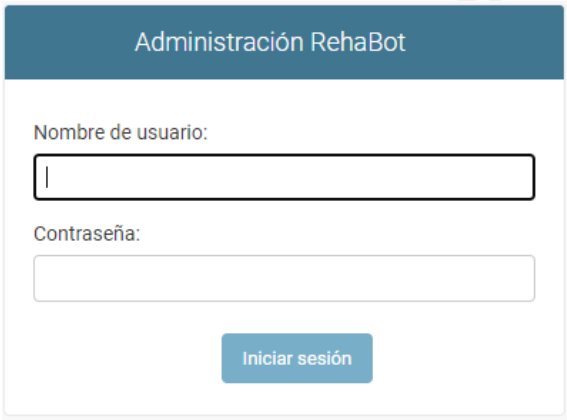

Imagen 44. Ventana de inicio de sesión en el panel de administración.

Una vez hayamos iniciado sesión accederemos a la página principal del panel de administración en la que se nos mostrarán las opciones a las que podemos acceder, pero en este caso nos vamos a centrar en nuestro caso de uso.

Imagen 45. Acceso a las secciones de Ejercicios, Pacientes y Sesiones desde el panel de administración.

Y como lo que queremos es añadir, deberemos pulsar en añadir. Como resultado de ello accederemos a la misma pantalla que hemos mostrado anteriormente.

Podemos crear tantos usuarios, ejercicios o sesiones como queramos, pero en el caso de las sesiones solo podremos enviar de una en una. Es decir, si quiero enviar dos sesiones a la vez el sistema me va a dejar, pero el usuario va a recibir la última puesto que el sistema nos sobrescribe los datos en el servidor y se queda con la última sesión.
Como solución tenemos la opción de crear muchas sesiones pensando a largo plazo, pero ir enviando las sesiones de una en una.

Caso de uso 2 - Editar pacientes, ejercicios o sesiones

Al igual que en caso de uso número 1, lo primero que debemos hacer es acceder a la ventana en la que queremos modificar algo, es decir, un paciente, una sesión o un ejercicio. Y podemos hacerlo por dos vías, por la aplicación web o a través del panel de administración.

El concepto es el mismo ya que para editar antes debe existir, por lo tanto, debemos de encontrar lo que queremos modificar.

Siguiendo esa filosofía, lo primero que haremos es buscar. En este ejemplo buscaremos un paciente en concreto.

1. En primer lugar, en el caso de que la lista sea muy larga, tenemos la opción de buscar y filtrar la búsqueda. Cuando hayamos encontrado lo que nos gustaría editar debemos de pulsar en “Editar Paciente”.

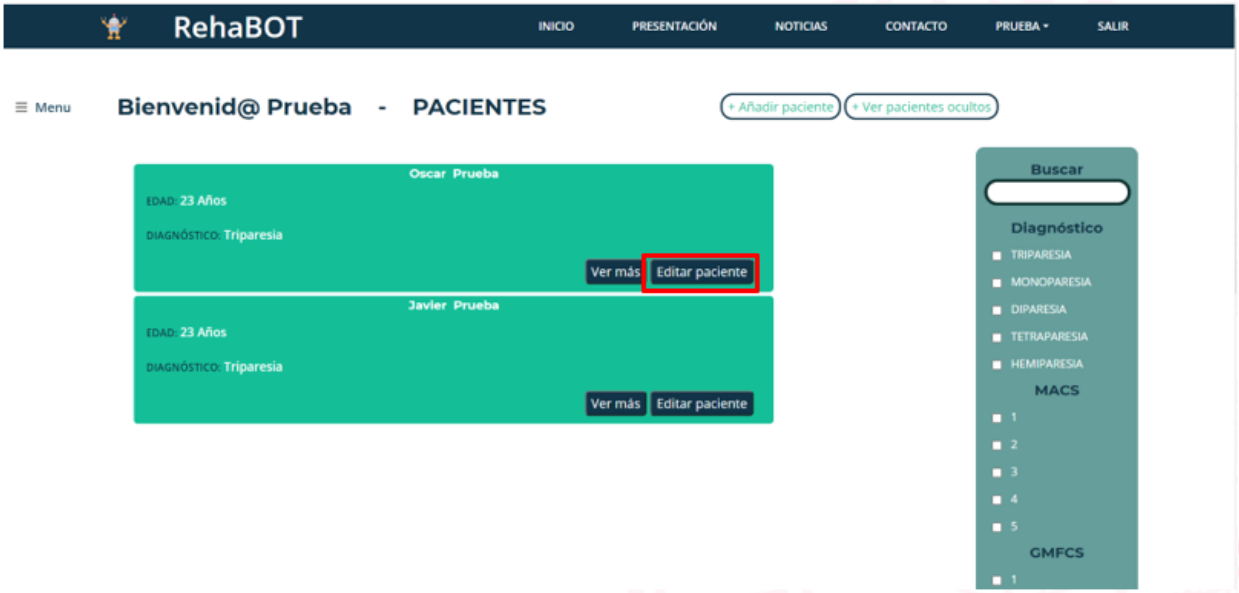

Imagen 46. Botón que nos permite editar los datos de un paciente.

2. Como ya hemos visto antes este enlace nos lleva directamente al panel de administración donde podremos editar los datos del paciente.

3. Existe otra posibilidad de hacerlo y es que, si en vez de estar en esa ventana decidimos primero ver más detalles sobre el paciente, accederemos a la siguiente pantalla, a través del botón "Ver más", donde también es posible editar el paciente.

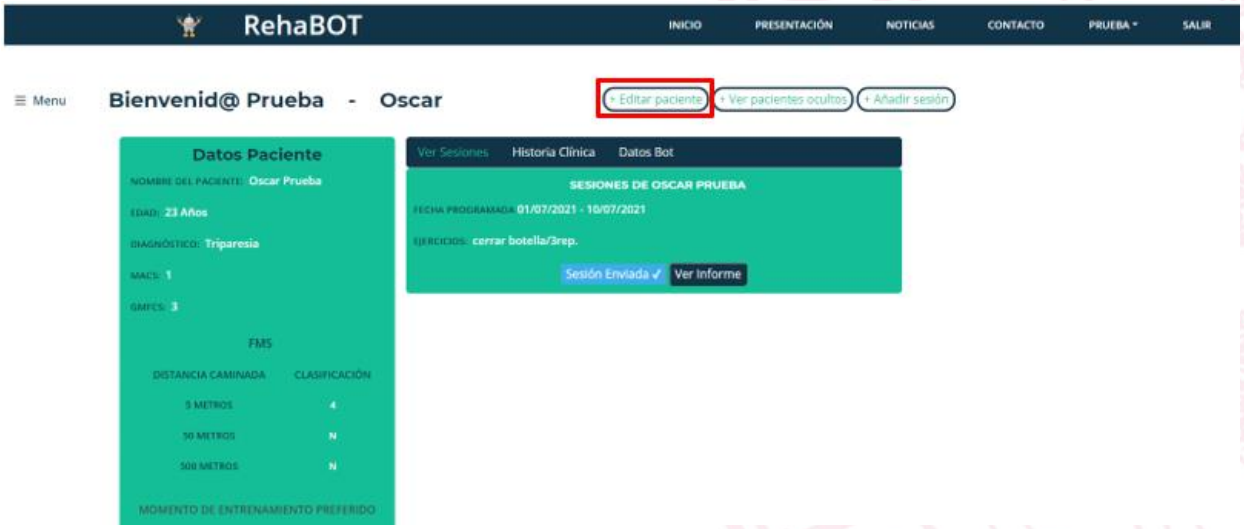

Imagen 47. Segunda opción de editar los datos de un paciente.

La diferencia entre crear y editar es que en la edición veremos los campos rellenos con los datos actuales y en la creación están vacíos esperando a ser rellenados. Una vez realizados los cambios se pulsa en "Guardar" y los cambios se habrán efectuado.

Al igual que en la creación podemos acceder directamente al panel de administración y desde allí seleccionar el paciente, ejercicio o sesión que queramos modificar.

Es muy importante tener en cuenta que, si queremos modificar una sesión, primero debemos comprobar que no haya sido ya enviada al paciente, porque si está enviada el paciente va a recibir la primera sesión. Hay dos maneras de solucionarlo:

1. Si la sesión se ha enviado por error, tenemos hasta las 23:59 de ese mismo día para modificarla y enviar la sesión rectificada. Tendríamos que acceder a la sesión, realizar los cambios y desmarcar la casilla de "Enviado". Después dentro de Sesiones > Enviar Sesión (La que hemos modificado).
2. Si la sesión se quiere modificar y han pasado de las 00:00. Lo que tenemos que hacer es crear una nueva sesión y a las 00:00 de ese mismo día se actualizará en el servidor y el usuario recibirá la nueva sesión al día siguiente.

Caso de uso 3 - Ocultar pacientes, ejercicios o sesiones

Dentro de la aplicación web existe una opción que nos permite ocultar elementos de la ventana principal de pacientes, ejercicios y sesiones. Lo que conseguimos es ocultar esos datos en la ventana principal si se da un caso excepcional, como, por ejemplo, un paciente que abandona el sistema tele-rehabilitación. Si después quisiera retomar este camino, no tendríamos que volver a dar de alta el usuario bastaría con desactivar la ocultación.

1. Accedemos a la ventana en la queremos ocultar elementos.
2. Seleccionamos el elemento que queremos ocultar.

Imagen 48. Acceso a los datos de un paciente para cambiar su estado de visible a oculto.

3. Ocultamos el elemento, en este caso, un paciente. Para ello desmarcamos la casilla "Sin ocultar".

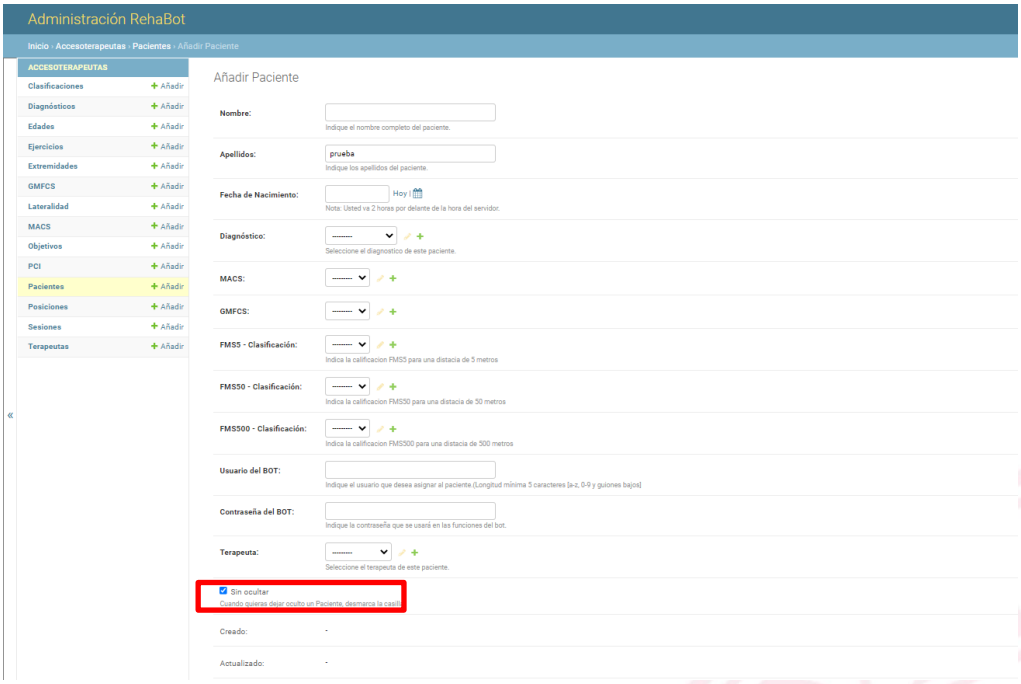

Imagen 49. Casilla que se debe desmarcar para ocultar a un paciente.

4. Después podremos consultar los pacientes ocultos desde el siguiente enlace.

Imagen 50. Botón desde el que podemos ver los pacientes ocultos que tenemos.

Observaremos algo similar a lo siguiente.

Imagen 51. Página que muestra un listado de los pacientes ocultos.

5. En el caso de querer volver a mostrarlo simplemente repetiremos los pasos, pero en este caso marcaremos la casilla “Sin ocultar”.

Caso de uso 4 - Consultar pacientes

Desde el menú principal de pacientes tendremos acceso para consultar la información resumida de nuestros pacientes. En el caso de querer obtener información más detallada tendremos que seleccionar el enlace “Ver más” del paciente que queramos.

Una vez dentro de la información detallada del paciente podremos consultar la siguiente información.

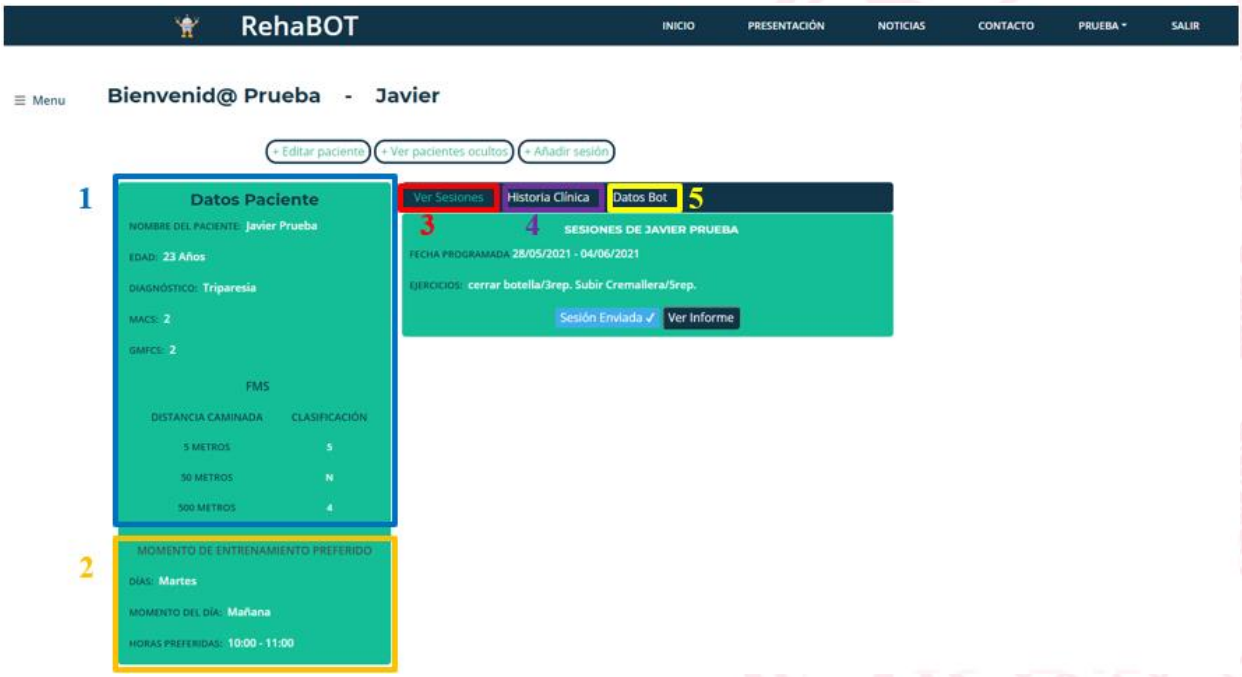

Imagen 52. Información detallada que se muestra de cada uno de los pacientes.

1 – Información detallada del paciente

2 – Momento preferido para realizar la tele-rehabilitación

3 – Sesiones asociadas al paciente, tanto las activas como las no activas en ese momento

4 – Historia clínica (En esta versión del sistema no está desarrollada)

5 – Usuario y contraseña asociados al paciente para usar en Telegram – Bot

Caso de uso 5 - Consultar sesiones

En el caso de que queramos consultar las sesiones que hemos programado a nuestros pacientes, podemos hacerlo desde el Menú principal > Sesiones > Consultar. Una vez dentro nos encontraremos con los siguientes datos.

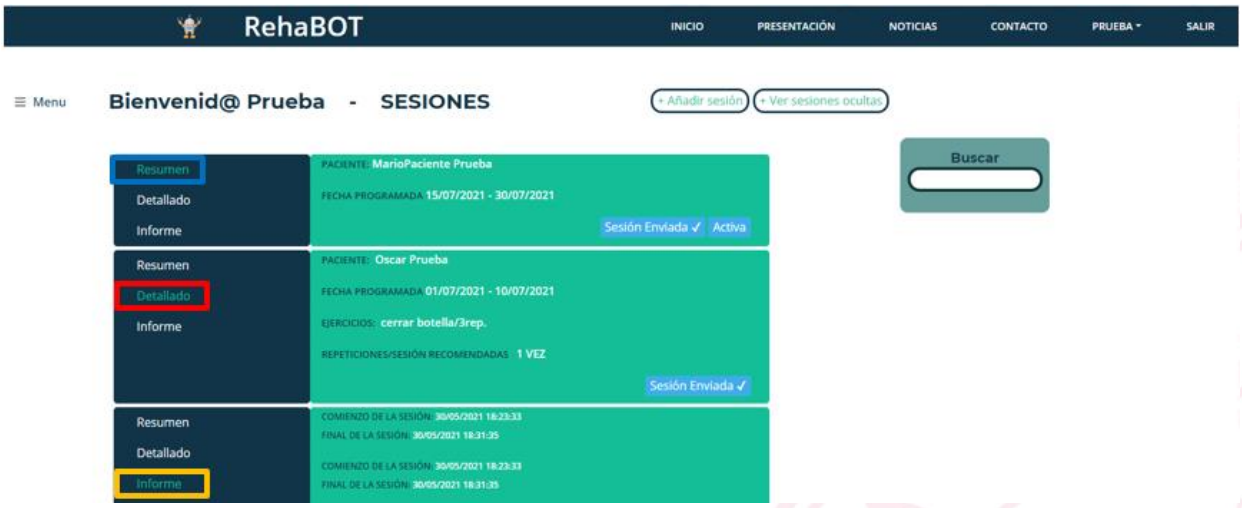

Imagen 53. Información de las sesiones divididas en tres secciones: resumen, detallado e informe.

Resumen – disponemos de la información resumida de la sesión: paciente al que se le asigna la sesión y fecha de inicio y final de la sesión.

Detallado – contiene la información resumida más los detalles de los ejercicios y repeticiones que se han seleccionado para ese paciente.

Informe – una vez la sesión ha sido completada, los datos recogidos por el bot se muestran en el siguiente orden: comienzo y final de la sesión, fecha y hora de cuando han sido realizados los ejercicios y las valoraciones de cada ejercicio.

Además, podemos consultar que sesión esta **activa** y cual no. Una sesión se encuentra activa si la fecha actual es superior a la fecha final de la sesión, es decir, si estamos dentro del plazo de tiempo para el que se ha programado esa sesión.

Caso de uso 6 - Consultar lista de ejercicios

El último tipo de consultas que podemos realizar es el de los ejercicios. Una vez dentro de la ventana de ejercicios podemos ver información resumida o información detallada de ellos.

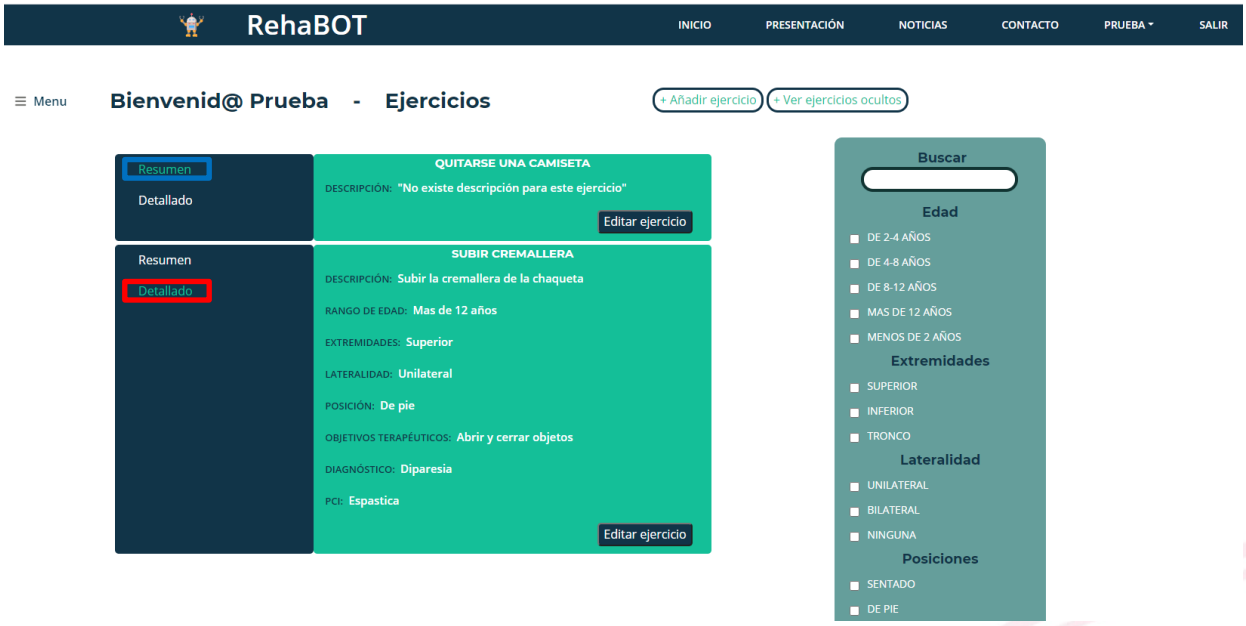

Imagen 54. Información de los ejercicios divididos en dos secciones: resumen y detallado.

Resumen – vemos el nombre del ejercicio y la descripción de este, si es que la tiene.

Detallado – a parte de la información resumida podemos comprobar el rango de edad, las extremidades, la lateralidad, la posición, los objetivos terapéuticos, el diagnóstico y la PCI que está relacionada con ese ejercicio.

Anexo 3 – Presupuesto total del proyecto

El coste total del proyecto se divide en dos partes. Por un lado, tenemos los costes asociados al *hardware* y por el otro los costes de desarrollo del *software*, en el que entraría tanto el desarrollo desde cero de los programas que se ejecutan como la configuración de *software* ya existente del que la aplicación hace uso. Empezaremos por el *hardware* detallando las características de cada uno de los componentes que han sido necesarios para el desarrollo de la aplicación como se muestra en la **tabla 22**.

Tabla 22. Coste de los componentes *hardware* de la aplicación.

Componente	Precio (sin IVA)	Precio (con IVA)
Raspberry Pi 4 Model B+ con 8 GB de RAM	88,02 €	106,50 €
Tarjeta micro-SD de 32GB	4,13 €	4,99 €
Caja oficial para Raspberry Pi 4	4,88 €	5,90 €
Fuente de alimentación de 5.1V y 3A	7,02 €	8,49 €
Pantalla LCD de 7 pulgadas	61,12 €.	73,95 €
Total	165,17 €	199,83 €

En este tipo de proyectos el coste es muy variable según los requisitos que tienen la aplicación que hay que desarrollar. Por lo general no hay un precio estándar establecido por hora entre desarrolladores de *software*, todo se basa en la complejidad del sistema a desarrollar. Durante el proceso de desarrollo de este proyecto se han usado tecnologías desconocidas para mí por lo que el desarrollo de la aplicación ha sido más un ejercicio de aprendizaje en muchas ocasiones que de desarrollo. Para el despliegue y programación de la aplicación web ha sido necesaria la realización de dos cursos sobre el uso y despliegue de aplicaciones web con Django. Además de la aplicación web, se ha desarrollado íntegramente el bot usando la API de Telegram y los correspondientes scripts tan necesarios para el correcto funcionamiento del bot. Esos scripts a su vez hacen consultas a un "bucket" de Amazon en el que tenemos almacenado los datos de las sesiones y el *feedback* de los pacientes, entre muchas otras cosas. El coste de mantenerlo en Amazon depende de las consultas y de los datos que se almacenen.

Por otra parte, Amazon nos mantiene a instancia que sirve de servidor para nuestro sistema. Por ello es necesario tener también muy en cuenta los costes asociados al mantenimiento de los servidores y el almacenamiento de los datos.

Una vez resumidos los sistemas *software* de los que depende la aplicación es importante hacer una estimación de los costes asociados al desarrollo, despliegue y mantenimiento de la aplicación como se observa en la **tabla 23**.

Tabla 23. Coste aproximado del desarrollo *software* de la aplicación.

Software	Trabajos incluidos	Tiempo*	Coste total**
Amazon Web Services	Configuración y puesta en marcha de Bucket S3	1 semana	1800 €
	Configuración y puesta en marcha de EC2	1 semana	1800 €
Desarrollo del bot	Bot, scripts, configuración y puesta en marcha de Raspberry Pi	1 mes y medio	7200 €
Desarrollo de la aplicación web	Desarrollo, despliegue y puesta en marcha	5 meses	36000 €
Formación	Cursos de formación sobre IT	1 mes	0€
	Total		46800 €

*Los valores asumidos en la tabla son que un mes se compone de 20 días laborables y cada día se trabajan 8 horas.

** Basándome en el mercado actual el coste por hora de un ingeniero junior se estima en 45€ de media.

Por lo tanto, el presupuesto final aproximado del proyecto realizado según estimaciones del coste por horas de un ingeniero junior es de **46999,83 €.**