industriales
etsii

Escuela Técnica Superior de Ingeniería Industrial

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Escuela Técnica Superior de Ingeniería Industrial

Herramienta de adquisición de parámetros fisiológicos relacionados con el nivel de ansiedad durante el tratamiento de los trastornos de ansiedad mediante terapia de exposición y prevención de la respuesta

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Autor: **Pérez Hernández, Diego**

Director: Pavón Pulido, Nieves

Codirector: López Riquelme, Juan Antonio

Cartagena, 06 de abril de 2021

Universidad Politécnica de Cartagena

En este trabajo se ha desarrollado una herramienta especializada para profesionales del ámbito de la psicología, concretamente, para que el terapeuta pueda tener una mayor información del estado físico y mental del paciente de una forma rápida, sencilla y con herramientas vía online.

Para que esto se pueda llevar a cabo, se ha implementado una herramienta de adquisición de datos con la que, a partir de ciertos sensores, se adquiere información que luego se almacena en una base de datos.

También, se ha desarrollado una aplicación web con la que se puede interactuar con distintos servicios los cuales van desde la visualización de los datos hasta la creación de diferentes entidades.

Además de los sensores que determinan ciertos parámetros fisiológicos del paciente, se han usado diferentes modelos de IA para poder medir parámetros del rostro del paciente gracias a una captura en formato vídeo desde la webcam del ordenador del paciente.

ÍNDICE

ÍNDICE DE ILUSTRACIONES

1. INTRODUCCIÓN.

En la actualidad, la cantidad de personas que sufren algún tipo de trastorno psicológico es muy alta, y dicha cantidad va en aumento. Estos trastornos pueden perjudicar al individuo en una gran cantidad de aspectos de su día a día, por ejemplo, en el ámbito económico, social, y laboral entre muchos otros. Para ayudar en dichos trastornos, en la práctica clínica, el problema se puede abordar desde distintos puntos de vista combinados o no entre sí. Concretamente, el terapeuta usa la información que le va proporcionando el paciente para así poder hacer un estudio subjetivo de su estado actual y la evolución que vaya consiguiendo a lo largo de la terapia. Dichos estudios podrían llegar a estar sesgados debido a que la información que se le proporciona al terapeuta se da desde un punto de vista subjetivo, por lo que sería interesante tener la capacidad de tomar medidas y así cuantificar de manera objetiva el estado del paciente para poder combinar dicha información con las opiniones y apreciaciones subjetivas.

Además, dada la situación en la que vivimos en la actualidad, supone un problema añadido la interacción presencial entre paciente y terapeuta que se tiene con las terapias que, normalmente, se practican en la actualidad, ya que, para que el paciente se sienta más cómodo y exprese con la mayor veracidad posible el estado en el que se encuentra, las terapias se ejecutan en su mayoría de forma presencial, y si se mandan ejercicios o pautas a seguir para el paciente durante su día a día no se suelen tener los medios suficientes como para efectuar un seguimiento de los mismos.

En relación con este tema, cabe destacar que podemos encontrar una gran variedad de herramientas que nos permiten hacer una evaluación de nuestro estado psicológico de forma más o menos fiable por medio de aplicaciones de móviles u online, o por relojes inteligentes que pueden indicar tu estado anímico, como el Apple Watch. Estas herramientas tienen la capacidad de facilitar en gran medida el tratamiento psicológico, ya que se puede hacer un seguimiento de una forma telemática disponiendo de las herramientas necesarias para poder tomar los parámetros fisiológicos. El desarrollo de una plataforma online que haga uso de distintas herramientas de adquisición de datos y las represente en dicha plataforma, de manera que sea de fácil acceso, con datos representativos y formato no presencial es por tanto, el objetivo del proyecto.

1.1. Objetivos.

En este proyecto, se pretende desarrollar una herramienta que pueda llegar a trabajar de forma online que funcione como herramienta de adquisición de datos, es decir, una herramienta en la que se visualicen, comparen y usen datos adquiridos de sensores, o en este caso, de herramientas virtuales que hagan de estos propios sensores. Aparte de los fisiológicos, se quiere poder interpretar datos del estado anímico del paciente.

También, se pretende tener una base de datos en la que tener todo tipo de información, ya sea personal, de los datos fisiológicos y del estado anímico. Toda esta información se usará para interpretar con mayor facilidad la condición del paciente.

Usar datos provenientes de sensores que analicen el estado del paciente frente a distintos estímulos. Estos sensores medirán distintos parámetros fisiológicos del paciente para que más adelante, el terapeuta, pueda analizarlos y obtener información cuantitativa del estado del paciente.

Por último, toda esta herramienta de adquisición de datos tendrá consigo una interfaz gráfica, en la que se representarán de forma estructurada y simple todo tipo de información necesaria, y la cual, incluirá también distintas páginas, opciones y la posibilidad de iniciar y cerrar sesión.

Para poder llevar a cabo todos estos objetivos se han desarrollado los siguientes subobjetivos:

- Generación de un modelo de datos en el que se crearán distintas entidades. La necesidad de este modelo de datos se da porque son necesarias muchas entidades relacionadas entre sí y cada una con características particulares.

- El diseño de la página web con la que el usuario puede interactuar de forma sencilla. Esta página web se ha diseñado para que el usuario pueda interactuar con el programa con todos los servicios y es la unión con todo el backend.

- El uso de dos modelos de IA con los que medir el estado anímico del paciente. Para conceder mayor información al terapeuta a parte de el estado fisiológico, se usan dos modelos que detectan el rostro, entre otros parámetros, que dan una información necesaria al terapeuta en cuanto a como reacciona el paciente a distintos estímulos.

- Analizar los resultados de los dos modelos de IA para poder obtener conclusiones de ventajas e inconvenientes de cada uno de ellos y cual funciona mejor en distintas situaciones.

1.2. Descripción de los capítulos.

Las secciones que se desarrollarán en este trabajo son Estado del arte, Materiales y Métodos, Diseño del Sistema, Conclusiones y Trabajo futuro.

- Estado del arte desarrolla la actualidad de las aplicaciones que ayudan a la salud mental y dispositivos que miden diferentes parámetros que ayudan a estar al tanto de tu estado mental actual. También, se trata la

actualidad de las nuevas herramientas como lo son la IA y el Big Data, herramientas muy útiles para muchos campos, entre ellos la visión artificial.

- Materiales y Métodos desglosa todo lo necesario para desarrollar trabajo. Estos materiales necesarios incluyen las herramientas usadas como lo son los entornos de programación, las librerías y los modelos usadas durante el trabajo

- Diseño del Sistema engloba todo el desarrollo técnico, como lo es el mapa de la base de datos, las clases, las funciones implementadas y el desarrollo de la interfaz gráfica.

- Por último, en la conclusión, se hace un breve resumen de todo lo implementado y ventajas e inconvenientes de las herramientas usadas durante el desarrollo del proyecto.

- Con los trabajos futuros, se explican posibles actualizaciones o mejoras que se podrían hacer para seguir implementando el trabajo. Estas mejoras van desde la utilización de nuevas herramientas, mejoras gráficas, implementación de sensores reales o el uso de nuevos modelos.

2. ESTADO DEL ARTE

Actualmente hay diversas aplicaciones online tanto para ordenador como para móvil, y gadgets portátiles o smartwatch. Uno de los gadgets que normalmente se usan es iRelax. iRelax es un dispositivo que controla tus constantes como lo son la respiración y pulso. Dicho gadget se utiliza principalmente para tener cierto control de estrés y controlar la respiración, pero dadas sus características también se podría usar para lo que a este trabajo le compete, que es el control vía online de la evaluación psicológica de los pacientes. Los smartwatchs son el dispositivo más usado que también tienen la capacidad de hacer cierto análisis de nuestro estado psicológico, como por ejemplo lo es el Samsung Galaxy Watch Active. Tanto este smartwatch como muchos otros no indican con claridad el algoritmo que utilizan para determinar el nivel de ansiedad del usuario. Estos smartwatchs tienen una eficacia relativamente buena, pero su evaluación psicológica solo se basa en las mediciones cardíacas, como lo pueden ser el ritmo cardíaco y/o la presión arterial.

En cuanto herramientas online, podemos encontrar todo tipo de test de autoevaluación emocional y psicológica por todo internet como el que nos ofrece el apartado de Bienestar Emocional del Ministerio de Sanidad del Gobierno de España [1] el cual nos proporciona varios test para evaluar diferentes apartados psicológicos como son estrés, ansiedad y depresión entre otros, y a partir de un umbral nos recomienda el que hacer. También, de manera online, se pueden encontrar multitud de vídeos, artículos y páginas web con numerosas técnicas de relajación y control tanto del estrés como de la ansiedad. Por ejemplo, una de las técnicas más usadas se trata del mindfulness. Como se indica en el artículo [2] tiene diversas definiciones, pero todas ellas tienen en común de que dicha técnica se basa en centrarse en el presente. En este mismo artículo, también indica las aplicaciones clínicas de dicha técnica.

Los móviles tienen una gran cantidad de aplicaciones para la relajación y meditación como Pacifica de Sanvello Health Inc. la cual es una de las aplicaciones más usadas, o Self-help Anxiety Management que es una aplicación desarrollada por la Universidad de Bristol. Estas, son dos ejemplos, pero hay muchas otras aplicaciones para meditar, yoga, mindfulness y similares las cuales ayudan a reducir el estrés y la ansiedad. Al contrario que las aplicaciones de relajación y limitación del estrés y la ansiedad, las aplicaciones para evaluar nuestro estado psicológico no son tan abundantes. Esto se debe principalmente al hardware de los propios móviles, ya que actualmente, la mayoría de ellos no tienen la capacidad de hacer medidas fisiológicas, se ven necesitados de una herramienta externa como lo puede ser un smartwatch. Aun así, móviles como el Samsung Galaxy 5 junto con la aplicación S Health sí que disponen de un medidor de frecuencia cardiaca como herramienta para evaluar el estrés

Recursos más específicos y técnicos en los que se pueden apoyar para determinar el estado anímico de una persona, son con el uso de herramientas de Inteligencia Artificial y Big Data. Usar una herramienta de inteligencia artificial para que se encargue de una tarea concreta, es un recurso muy útil y que cada vez está más al alza. El principal problema que tiene esta herramienta es el entrenamiento necesario

para que trabaje con una precisión optima, y para dicho entrenamiento es necesario una gran cantidad de datos y recursos, servidores, procesadores de alta gama y demás. Pero aún así, siempre se puede desarrollar una inteligencia artificial que reconozca caras y que reconozca el estado anímico de la persona con mayor o menor precisión. Si no se tienen los recursos o la capacidad necesaria para desarrollar una inteligencia artificial de dichas magnitudes, empresas como Google ofrecen dichas herramientas al consumidor. Google, con su herramienta de Cloud Vision Api, a partir de una imagen da la posibilidad de detectar caras, darnos la posición de puntos clave de la misma y muestra la emoción que tiene en la imagen entre: alegría, ira, tristeza y sorpresa, y con la precisión que tiene de cada una de ellas.

El trato de las emociones de cada uno es algo complejo de definir y categorizar. En el artículo [3] se indica que una de las formas que tiene la psicología para describir las emociones, se basa en la evaluación, es decir, si se trata de algo negativo o positivo, y el nivel de activación, es decir, factores que pueden definir la situación. También, cada sentimiento produce variaciones biológicas en el organismo como lo pueden ser, cambios en la expresión de la cara, aumento o disminución de la temperatura corporal, sudoración, aumento o disminución del ritmo cardiaco, y cambios en la actividad cerebral entre muchos otros. Estos cambios biológicos están muy bien documentados en diferentes artículos y libros, pero tienen la problemática de reducir en exceso la valoración real del sentimiento a tratar.

La inteligencia artificial avanza a gran velocidad gracias a que tienen una comunidad muy amplia que comparten continuamente sus trabajos entre unos y otros. Actualmente, está al alza el concepto de Nigromancia Digital, el cual se basa en el uso de inteligencias artificiales para sustituir la cara de una persona realizando cierta acción, por otra, y en el caso concreto de la Nigromancia Digital, de personas ya fallecidas. Con esto se puede conseguir que parezca que la persona todavía sigue viva.

La Nigromancia Digital, está muy unida a la detección de caras ya que usan técnicas de entrenamiento de inteligencias artificiales muy similares. Actualmente, la técnica que se usa para el procesamiento y entendimiento de imágenes y videos se llama Autoencoder. Los Autoencoders, son redes neuronales que se entrenan a raíz de que se le suministra una imagen de entrada y la red debe de dar por salida la misma imagen, pero durante el proceso, la imagen de entrada se comprime mucho, hasta el punto que si la red neuronal quiere mostrar por salida lo que se desea, debe de aprender a buscar patrones. Con estos patrones, la inteligencia artificial puede determinar facciones de la cara, rostros completos, objetos o para lo que haya sido entrenado la red. También, una vez determinados los patrones de un rostro, se puede entrenar una inteligencia artificial incluso para que determine la emoción que hay en la imagen, o incluso simularla.

3. MATERIALES Y MÉTODOS

En los siguientes apartados, en primera instancia, se desglosarán las diferentes herramientas de las cuales se ha hecho uso durante el desarrollo del trabajo, y a continuación, se detallará la metodología empleada para hacer frente cada uno de los problemas.

En cuanto a la estructura de los apartados, tanto en la descripción de las herramientas empleadas como en el apartado de la metodología usada se hará de forma cronológica, es decir, conforme se han ido encontrando los diferentes problemas y/o se ha querido ir desarrollando. Se seguirá este orden para mayor comprensión del desarrollo del propio trabajo. Si en algún caso no se siguiera este orden, porque así convenga, se dejará indicado.

3.1. Herramientas usadas

Las herramientas que se ha visto necesario usar durante el desarrollo del trabajo son principalmente herramientas online y/o software, principalmente plataformas para programación en diferentes lenguajes como Java, JavaScript, HTML5 y C++ entre otros, con sus respectivas bibliotecas más o menos específicas, necesarias para el desarrollo del trabajo. Las bibliotecas fuera del uso común, que se haya necesitado usar para el desarrollo del trabajo se explicarán en este apartado.

3.1.1. Eclipse

Se trata de una plataforma para programar diferentes tipos de aplicaciones. Este software es multiplataforma y gratuito. En él se programará la herramienta que se utilizará como interfaz gráfica del proyecto, las distintas clases que se usarán y los distintos endpoints, es decir, hacer peticiones a un servicio web que a raíz de unos valores de entrada nos dará otros valores de salida. También, tiene una herramienta de depuración bastante potente, para poder localizar y arreglar los diferentes errores.

En el caso de este proyecto, en Eclipse se programará en lenguaje Java, JavaScript y HTML5.

3.1.2. Postman

Es una herramienta cuya principal función es dar la posibilidad de hacer peticiones sobre distintas API REST con peticiones bajo el protocolo de http y https. Postman se usará para hacer las peticiones a los diferentes endpoints que se crearán para ejecutar distintos algoritmos, guardar información en la base de datos o que nos devuelva cierta información.

Para hacer uso de ella solamente es necesario saber la dirección de la petición a la que queremos acceder, y los datos que queremos subir a dicha petición. En este proyecto se usará el formato JSON [4] tanto para recibir como para enviar la información a las distintas peticiones.

3.1.3. Google Cloud SDK

Es una de las muchas aplicaciones que se encuentran dentro de la Google Cloud Platform. Esta, hace uso de multitud de herramientas para administrar tanto aplicaciones como recursos de Google. El SDK también puede ser instalado en primera instancia en la herramienta de Eclipse, citada en el apartado 3.1.1. del proyecto. Esta aplicación la ejecutaremos en la línea de comandos y también la instalaremos desde la misma.

Para este proyecto, se hará uso de la herramienta de base de datos que contiene esta aplicación. Para ello, solamente se deberá de tener abierto un proyecto en Eclipse el cual esté vinculado a un proyecto creado en la Google Cloud Platform. Una vez hecho eso, tener tanto instalado como inicializado el SDK en el ordenador y ejecutar el siguiente comando:

```
gcloud beta emulators datastore start
```

Y ya se podrá usar esta herramienta perteneciente a la aplicación de Google Cloud SDK como la base de datos que se usará en el proyecto. Siempre que se quiera acceder a dicha base de datos, ya sea para guardar, extraer o borrar información, se deberá ejecutar el comando anterior y mantener abierta el símbolo de sistema en el que se ha ejecutado dicho comando.

3.1.4. Objectify

Objectify [5] es una biblioteca específica para lenguaje Java, la cual, está específicamente diseñada para trabajar con el almacén de datos que se usa con Google App Engine. En el caso de este trabajo, la base de datos es el emulador de Google Cloud SDK que se explicó en el apartado 3.1.3.

Los puntos principales que hacen destacar la biblioteca de Objectify, y también por los que se ha escogido dicha librería y no otra son: fácil comprensión, gran rendimiento, gran variedad de funciones intuitivas, capacidad de modelar estructuras de datos sofisticadas, estructura de modelo de datos distribuidos, puede subexistir con diferentes herramientas de almacenamiento de datos, puede hacer variaciones en tiempo real y tiene un nivel de abstracción óptimo.

3.1.5. Cloud Endpoints

Google Cloud Endpoints [6] es una herramienta para el desarrollo y monitorización de las diferentes APIs dentro del entorno de Google Cloud. Con esta herramienta se puede acceder a las distintas aplicaciones y estar todas ellas protegidas mediante una autentificación para los usuarios. También sirve tanto para aplicaciones web como para móviles.

Los endpoints serán la herramienta principal durante gran parte del trabajo. Gracias a ellos se pueden hacer llamadas a distintas funciones del propio programa mediante enlaces http. Con los endpoints, también se pueden administrar las diferentes APIs del entorno de Google Cloud Platform.

En cuanto a la programación de estos endpoints en la plataforma de Eclipse, esta ya tiene la librería incluida para poder trabajar con ellos sin necesidad de descarga externa gracias a las Google Cloud Tools de dicha plataforma.

3.1.6. Material Design

Es un sistema desarrollado por Google que dota de diferentes herramientas para el diseño web. Principalmente se centra para el diseño de aplicaciones Android pero también se puede usar para desarrollo web, iOS y Flutter.

En este sistema se pueden encontrar variedad de iconos, diseño de botones, barras de aplicaciones y tarjetas entre otros, al igual que toda la información necesaria para poder trabajar con todos los diseños [7]. Esta gran variedad de herramientas es de utilidad para hacer una interfaz gráfica de la aplicación web más limpia, clara e intuitiva.

Esta herramienta es necesaria para el diseño de la aplicación web ya que mejora estéticamente el diseño de la misma y proporciona diseños que hacen que resulte fácilmente entendible la navegación dentro de la aplicación web.

3.1.7. Google Charts

Google Charts [8] es una herramienta desarrollada por Google que proporciona una forma para poder visualizar gráficos en las interfaces gráficas de aplicaciones web. Esta herramienta tiene multitud de diferentes gráficas a los que se puede optar y facilidad para poder personalizar cada gráfico.

Normalmente, para utilizar Google Charts, se usa el lenguaje de programación de JavaScript, y para poder utilizar la galería de gráficos, hay que descargar la librería que proporciona Google. Todas las gráficas necesitan una tabla de datos, la cual se puede usar accediendo a una base de datos que soporte la herramienta o diseñando la propia tabla de datos en el propio script.

Para representar gráficamente los datos de los sensores para facilitar la visión de los mismos al terapeuta, se puede hacer mediante JavaScript y HTML5, pero esto resulta muy poco intuitivo y no es lo más adecuado estéticamente. Por ese motivo se ha escogido esta herramienta, ya que facilita en gran medida el graficar la información necesaria de forma clara y con gran variedad de modificaciones y opciones en las que elegir entre los distintos tipos de gráficas con las que se tiene acceso.

3.1.8. Code::Blocks

Se trata de un entorno para programación de código abierto. En él tiene una gran variedad de compiladores y el lenguaje de programación que se suele usar en este entorno es C o C++.

Es uno de los entornos de programación para C y C++ más usados ya que contiene todas las bibliotecas comunes y añadir otras nuevas resuelta sencillo. Es multiplataforma y gratuito [9].

Ha sido necesario el uso de otro entorno de programación como lo es Eclipse, porque se ha requerido de simular los datos de los sensores. Para ello, se necesitaba de otro entorno especializado en código C y C++ ya que es con el que más sencillo resulta esta tarea, sobre todo al necesitar enviar mensajes de tipo POST y existir ya un lenguaje de programación llamado CURL que puede enviar este tipo de mensajes y está muy relacionado con C y C++. De entre todos los entornos que pueden hacer lo anteriormente dicho, se ha escogido Code::Blocks por ser con el que más familiarizado se está.

3.1.9. Libcurl

Es una librería para el lenguaje de programación C y C++. Esta librería se trata de un intérprete de comandos del lenguaje de programación Curl. Curl, se basa principalmente en un lenguaje orientado a objetos diseñado principalmente para aplicaciones web.

Libcurl pretende ser una librería para ayudar con la transferencia de datos no supervisados. Al soportar certificados como POST, HTTPS y HTTP entre otros, lo hace idóneo para trabajar con los endpoints que se usan a lo largo del desarrollo del trabajo.

Esta librería se puede incluir en el entorno de programación de Code::Blocks, y por ello se utilizará para que puedan interactuar el programa que se desarrollará en C con los endpoints programados Java en el entorno de Eclipse.

3.1.10. Google Cloud Vision API

Es una de las herramientas que pertenece a Google Cloud Platfom. Esta aplicación se basa en dar herramientas para que los desarrolladores integren funciones de detección de imágenes.

Cloud Vision API tiene una variedad de herramientas en las que se encuentran: detección de puntos de referencia, detección de rostros, detección de texto, reconocimiento de contenido explícito.

Esta aplicación consume muchos recursos para Google, por lo que es necesario pedir una autorización para poder hacer uso de esta aplicación. Para conseguir esta autorización, previamente, es necesario crear un proyecto en la consola de Google Cloud. Para dicho proyecto se debe de habilitar el uso de la aplicación de Google Cloud Vision API, y tras configurar el proyecto se puede acceder a las autorizaciones de uso de la aplicación. Estas autorizaciones pueden ser de diferentes tipos, que van desde tokens (códigos alfanuméricos) y vinculación con una cuenta. En el caso de este proyecto, se han usado los tokens, los cuales hay que introducir en el endpoint que llame a esta aplicación. De esta forma, solo quien tenga acceso a los tokens, tendrá acceso a la aplicación que se está usando.

Una vez hechos los requisitos previos, se pueden utilizar diferentes métodos para acceder a la aplicación como lo son el uso de bibliotecas, mediante línea de comando o POST.

Para enviar la información a la aplicación de la imagen a tratar, el tipo de detección a usar y demás configuración se usa siempre el formato JSON. Y para que la aplicación reconozca la imagen a tratar, se debe codificar a Base64 o almacenarlo en una base de datos de Google.

La respuesta que nos proporciona la aplicación viene siempre en formato JSON. En este JSON que envía la aplicación como respuesta aparecen multitud de parámetros que dependerán del tipo de detección que se use en ese momento.

Para este trabajo se usará concretamente la herramienta de análisis facial. Dicha herramienta indicará multitud de puntos de referencia entre los que destacan:

- LEFT_EYE
- RIGHT_EYE
- MIDPOINT_BETWEEN_EYES
- LOWER_LIP
- MOUTH_LEFT
- MOUTH_RIGHT
- MOUTH_CENTER
- LEFT_EYE_RIGHT_CORNER
- LEFT_EYE_LEFT_CORNER

- RIGHT_EYE_RIGHT_CORNER

- RIGHT_EYE_LEFT_CORNER

- LEFT_EAR_TRAGION

- RIGHT_EAR_TRAGION

- detectionConfidance

- landmarkingConfidence

- joyLikelijood

- sorrowLikeligood

- angerLikelihood

- surpriseLikelihood

Entre todos estos puntos de referencia indicados, encontramos las referencias de la imagen 2D a analizar, puntos de la cara que hacen referencia a los ojos, boca, nariz, y orejas entre otros, y por último, nos da diferentes ángulos de la imagen, la confianza que se tiene tanto del rostro en general como de los puntos de referencia de la cara y una confianza para cada una de las expresiones que tiene en cuenta el modelo. Estas últimas confianzas, en el modelo tendrán un porcentaje de confianza igual que el resto de confianzas, pero el modelo tan solo nos devuelve la confianza en nivel cualitativo desde mucha seguridad hasta muy poca seguridad o incluso no se detecta nada.

Nótese que, para que este modelo detecte correctamente la expresión que muestra el rostro del sujeto, dicha expresión debe de estar muy marcada o forzada, y aun así, en ciertos casos no lo representa con mucha confianza, siendo la felicidad el sentimiento que muestra con mayor fidelidad.

Independientemente de estos últimos problemas, se trata de una herramienta muy potente que se puede usar como referencia, sobre todo en la parte de los puntos de referencia de la cara.

3.1.11. TensorFlow

TensorFlow es una biblioteca en lenguaje de Python que pretende dar recursos para el desarrollo de IA de todo tipo, ya sea centrada tanto en Machine Learning como en Deep Learning.

Alrededor de esta biblioteca se ha formado una gran comunidad en la que se comparten gran variedad de modelos. Los modelos de IA son programas ya codificados y entrenados para hacer su labor y dispuestos a ser comercializados. En la página oficial de TensorFlow [10] se puede encontrar tanto la librería como gran variedad de modelos de ejemplo. Aunque esta librería se centre principalmente en el lenguaje de

programación de Python, también se pueden encontrar modificaciones y modelos para otros tipos de lenguaje, como JavaScript.

En el caso de este trabajo, se ha usado como referencia un modelo de visión artificial, para el análisis facial. Con dicho modelo se analizará la cara del sujeto superponiendo una máscara al rostro del sujeto, la cual representa diferentes puntos faciales del sujeto. Concretamente, en este caso, el modelo da como resultado a tiempo real 486 puntos de la cara basándose en algoritmos predictivos del movimiento del rostro como se observa en la Ilustración 1.

La máscara que crea este modelo es muy fiable y se usa a lo largo del proyecto como apoyo a la otra herramienta usada para la detección de los rostros: Google Cloud Vision API. El modelo de TensorFlow es significativamente más fiable que la herramienta de Google, pero el modelo no tiene la capacidad de detectar los sentimientos del sujeto al contrario que la herramienta de Google.

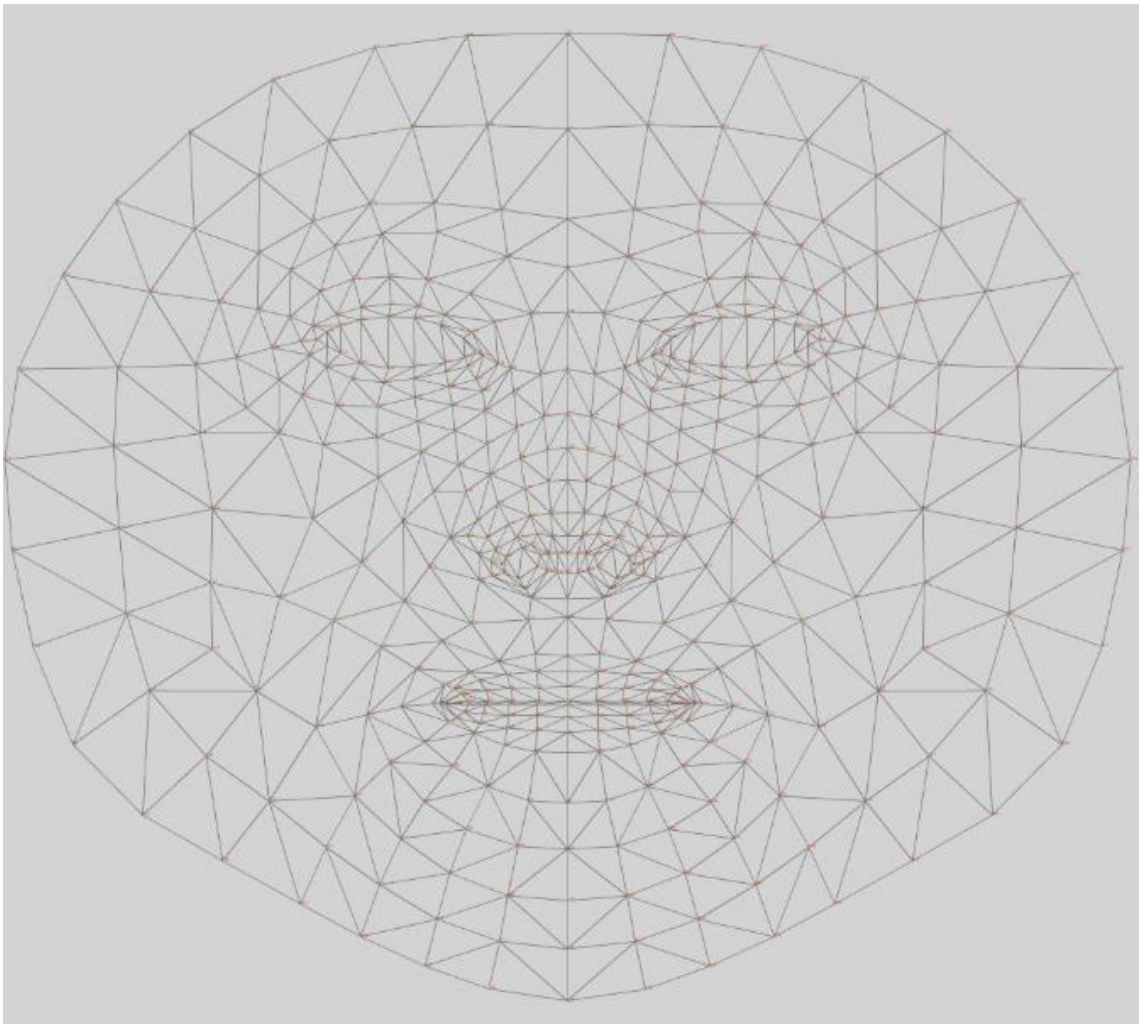

Ilustración 1Máscara generada por el modelo de TensorFlow

La Ilustración 1 representa un ejemplo de como se quedaría la máscara una vez se plasman las coordenadas que nos da como resultado los 486 puntos del modelo.

4. DISEÑO DEL SISTEMA

En este capítulo se reflejará el desarrollo del proyecto, los problemas que han ido surgiendo y como se han resuelto, ejemplos del código utilizado, y la explicación de los mismos.

El desarrollo del proyecto se puede dividir en tres puntos clave, que son: creación y uso de clases, desarrollo de los endpoints y programas adicionales e interfaz gráfica. También será necesario una breve explicación de la puesta en marcha del proyecto, es decir, unos preparativos previos para el correcto funcionamiento del programa.

Para enlazar todos los diferentes puntos del proyecto se ha elaborado un sistema como el que se observa en la ilustración 2:

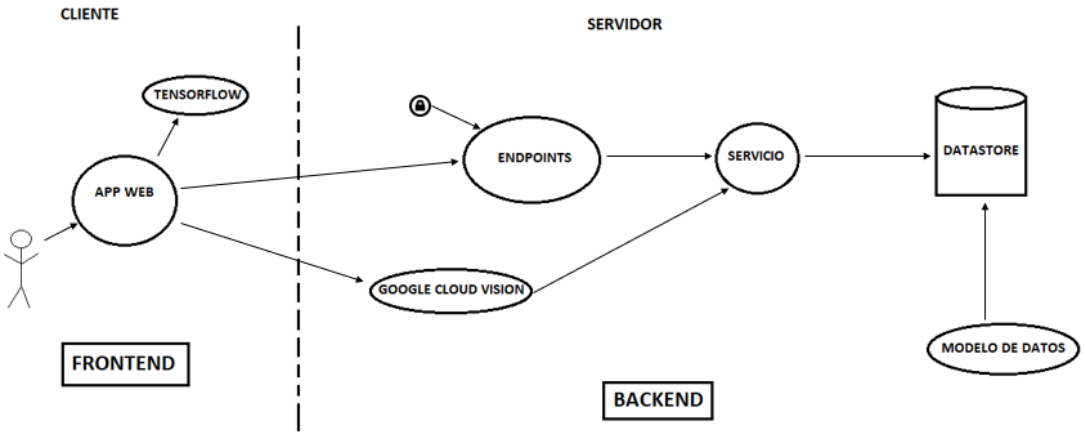

Ilustración 2 Diseño del sistema

En la ilustración 2 se dejan ver dos secciones como lo son el frontend y el backend. En el frontend se tienen las herramientas que interactúan directamente con el usuario, que en este caso sería la aplicación web y el modelo de TensorFlow. También podrían entrar herramientas como lo son la webcam o diferentes herramientas IOT.

En el backend se tiene la zona a la que el usuario no puede acceder de forma directa, es decir, el servidor. En el caso de este proyecto se encuentra la herramienta de Google Cloud Endpoints la cual llama a los servicios que son necesarios para el desarrollo del trabajo. Estos servicios también son llamados desde la aplicación de Google Cloud Vision API. Con los servicios, se accede a la base de datos para la cual se ha necesitado diseñar un modelo de datos debido a las diferentes interacciones entre los campos.

En los endpoints es donde se encuentra la zona de seguridad, ya que se necesita tener autorización para poder tener acceso a los distintos servicios con los que accedes desde los endpoints, siempre y cuando este lo requiera.

Todo el sistema trabaja de forma online, es decir, el usuario quien tenga acceso a la aplicación web se comunica con los endpoints cuando requiera de algún tipo de llamada a raíz de internet, con llamadas de tipo http y https.

Lo primero que se ha diseñado en el sistema es la base de datos, porque, aunque sea un tratamiento de izquierda a derecha, partiendo de lo que tiene acceso el usuario hasta la base de datos, es necesario tener perfectamente clara y estructurada la base de datos, y de ahí parte el modelo de datos.

4.1. Modelo de datos

La estructura de la base de datos que se usa es relacional, es decir, algunos tipos de objetos se pueden relacionar con otros, y por ello, se debe de diseñar previamente un modelo que explique estas relaciones, y eso es el modelo de datos.

Un modelo de datos es un tipo de lenguaje que se dedica a explicar una base de datos, es decir, las relaciones entre los distintos datos, el tipo de dato que son, la estructura que tienen los datos y demás.

El modelo de datos de este proyecto se podría representar como:

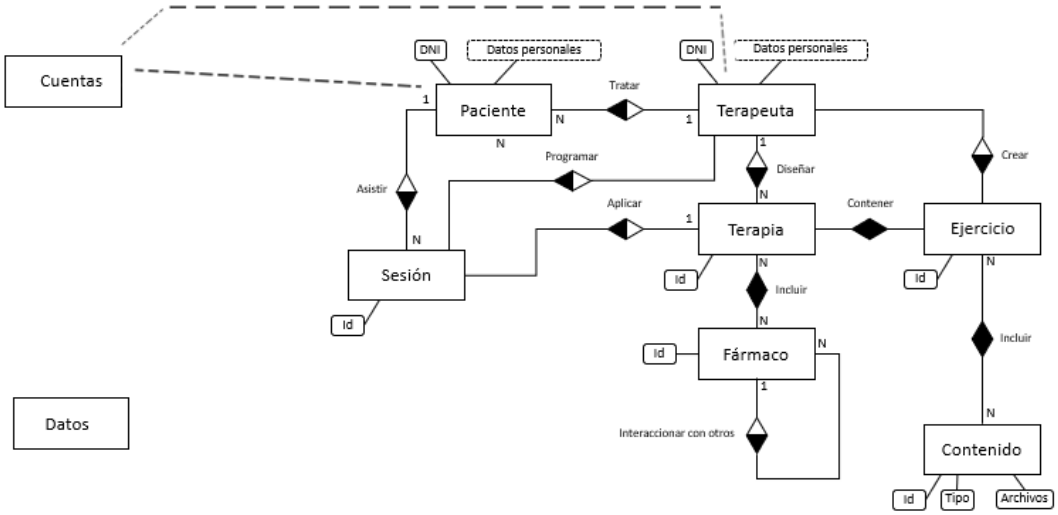

Ilustración 3Modelo de datos

En el caso de este proyecto, se pueden observar tres estructuras de datos diferentes que son: una estructura general centrada en las terapias, y otras dos estructuras de datos más simples, una centrada en los datos recogidos de las pruebas y la otra centrada en las cuentas de inicio de sesión.

El primero de estos modelos de datos, el que se centra principalmente en las terapias, tiene una estructura similar a la siguiente:

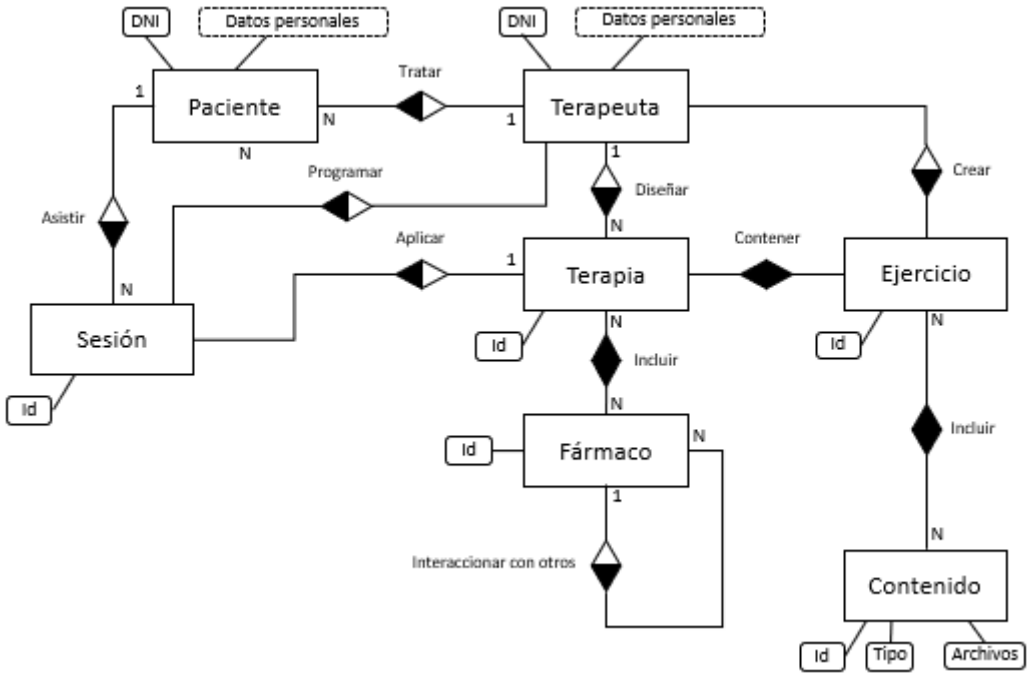

Ilustración 4: Modelo de datos centrado en las terapias

En la imagen, se pueden observar siete diferentes tipos de objetos:

- **Paciente:** Este objeto contiene la información referida al paciente del terapeuta. Estará compuesto por cada uno de los pacientes que estén en el proyecto independientemente del terapeuta al que estén asignados. Los datos que estarán dentro de cada objeto tipo paciente serán principalmente los datos personales de cada uno de ellos.

- **Terapeuta:** En esta clase se almacenan todos los diferentes terapeutas que contiene el proyecto. Cada uno de los objetos de tipo terapeuta almacenará la información personal de cada uno de ellos y una lista de pacientes que tiene dicho terapeuta.

- **Ejercicio:** En esta clase se almacenarán todos los diferentes tipos de ejercicios que los pacientes tienen acceso a hacer. Los ejercicios contienen una lista de contenidos a hacer por el paciente y una duración total del ejercicio a hacer.

- **Contenido:** Los contenidos son diferentes tipos de archivos, fotografías, vídeos, exámenes, ejercicios y demás que evalúan al paciente. Es decir, se trata de ejercicios que tendrán que hacer los pacientes durante las sesiones o en casa. La clase Ejercicio contiene uno o más contenidos que son los que tendrá que hacer el paciente durante las distintas sesiones.

- **Fármaco:** Existen muchos tipos diferentes de fármacos y cada uno sirve para un tratamiento en concreto. En este objeto estarán almacenados todos los fármacos que usan los terapeutas tanto para las sesiones presenciales como los que usan para

recetar a los pacientes para que se tomen en casa. También cada fármaco, por su naturaleza química, pueden interactuar de diferente manera entre ellos, por ese motivo, en esta misma clase también hay un apartado en el que se incluye la interacción que tiene con otros fármacos en el caso que tenga dicha interacción.

- **Terapia:** En este objeto es donde convergen la mayoría de los demás objetos. Es el tipo de objeto más importante y sobre el cual se han ido desglosando el resto de objetos. Terapia tiene un identificador y en ese identificador y para cada terapia hay asignado un ejercicio, un fármaco e información sobre la suministración del fármaco como lo es la cantidad, cada cuanto tiempo y durante cuantos días se debe de tomar el fármaco, y también, se hace referencia a que paciente se le aplica la terapia.

- **Sesión:** Se trata de un objeto que está estrechamente ligado con terapia. En este objeto se deberán asignar las sesiones que pertenecen a cada terapia asignándole una fecha, un paciente y una terapia.

  Todos estos objetos están implementados en el programa de forma individual y separada, teniendo así cada uno de ellos una clase asignada con su nombre.

  Como se ha explicado con anterioridad, se discernían 3 diferentes tipos de estructuras de datos, y la segunda de ellas es la que engloba la recogida de datos:

  Esta es la estructura de los datos más simple ya que solamente engloba a los datos y no tiene ningún tipo de relación con ninguna otra clase, excepto uno de los tres diferentes tipos de datos que es el que se encarga de almacenar los datos de los sensores.

  Dentro de la estructura tenemos tres diferentes clases de tipos de datos, los cuales dependen de la herramienta que se use para adquirirlos. Estas tres clases son:

  - **Datos de los sensores:** En esta clase se almacenan los datos que vienen de los sensores que dan información del estado fisiológico del paciente. Estos datos engloban las medidas de:

    -Pulsación: Es decir, con este parámetro se medirá el ritmo cardíaco del paciente.

    -Tensión: Se medirá la tensión del paciente durante el proceso de ejecución del ejercicio.

    -Galvánico: Este parámetro mide la cantidad de sudoración de una persona. Esto será útil de cara a detectar el nivel de ansiedad y nerviosismo del paciente.

  - **Datos de la aplicación de Google Cloud Vision API**: Aquí se almacenarán gran parte de los datos que nos suministra la aplicación de Google. Como se explicó con

anterioridad, la aplicación de Google nos da como salida multitud de parámetros. Muchos de estos parámetros no son necesarios para el desarrollo de este trabajo o son redundantes, y dada la gran cantidad de datos que se nos suministra, para evitar exceso de datos en nuestra base de datos solo se almacenarán unos pocos, los cuales son:

-Alegría.

-Tristeza.

-Ira

-Sorpresa

-Foto subexpuesta.

-Foto borrosa.

-Verosimilitud de la foto.

-Coordenada X del ojo izquierdo.

-Coordenada Y del ojo izquierdo.

-Coordenada X del ojo derecho.

-Coordenada Y del ojo derecho.

-Seguridad de las marcas

- **Datos del modelo de TensorFlow:** En esta clase se almacenarán los datos del modelo de TensorFlow que se usará para detectar la cara del paciente y los movimientos y gesticulaciones que hace el paciente durante el desarrollo de todo el ejercicio. Las fotos se quedarán almacenadas en la base de datos de forma numerada, desde la 1 hasta las que se hayan hecho durante el ejercicio, y tras la revisión de las fotos por parte del paciente y pasar a hacer un nuevo ejercicio, estas fotos se borrarán. Este proceso de borrado se hace debido a la gran cantidad de información que se almacena debido a la mascarilla.

La última estructura del modelo de datos es:

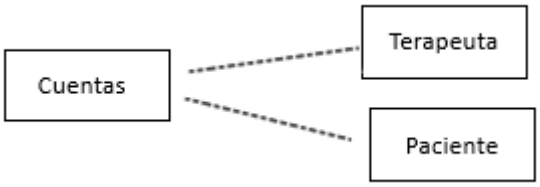

Ilustración 5Estructura de los datos de las cuentas

Esta estructura tiene cierta relación tanto con pacientes como con terapeutas, pero se trata de una relación indirecta de ahí que se separe en otro grupo distinto.

En esta estructura se almacenarán las diferentes cuentas que se pueden usar para acceder a la herramienta.

En cuentas se encontrarán dos tipos de clases:

- **Cuentas:** En la que se almacenarán todas las cuentas en las que estará representado el nombre, una contraseña y el rol, si es terapeuta o paciente. Esta clase es la que tiene la relación indirecta con los pacientes o los terapeutas ya que se debe de comprobar si el paciente y/o terapeuta existen antes de ser creadas sus respectivas cuentas.

- **Registro:** En esta clase se almacenará un registro de quien está usando o no la aplicación, junto con su fecha y hora de inicio de sesión para comprobar, si fuese necesario, alguna intromisión indeseada o ver si el paciente ha iniciado sesión para hacer sus ejercicios y, en el caso de este proyecto, se usará para comprobar el rol de quien ha iniciado sesión y así redirigirle a la página correcta, la del paciente cuando inicia sesión un paciente y la de terapeuta cuando inicia sesión un terapeuta.

4.2. Implementación del almacén de datos

En este apartado, una vez explicado el modelo de datos en el cual se ha dicho todos los datos tipos de clases que se quieren almacenar, se explicara lo que es una clase, donde la creamos y todas las clases que han sido creadas en el proyecto. También se explicarán todos los parámetros que hay en cada clase, la información que estos aportan, el tipo de dato que son cada parámetro y todos los métodos de la clase en cuestión.

Las clases, al igual que el documento java de endpointtfg y que los documentos de ObjectifyWebFilter y ObjectifyWebListener se incluyen dentro del paquete de java tfg.com. Estas clases también se tratan de documentos de tipo java y a lo largo del proyecto se diseñarán dos tipos de clases diferentes. Una de estas clases será las que se usarán para el almacenamiento de los datos, es decir, en la base de datos se encontrará

una clase que será esta misma que se diseñe con sus respectivos parámetros. El otro tipo de clase será para el uso de listas de objetos. De esta última no se crearán objetos para almacenarlos en la base de datos, sino que se usará para coger objetos de otras clases y hacer listas de ellos. Estas listas no se requerirán que se almacenen en la base de datos ya que solo se usará para captar lista de datos ya almacenados.

Para la implementación de las entidades, como en cualquier otro lenguaje de programación, es necesario en primera instancia añadir todas las librerías pertinentes. Además, las variables que están dentro de cada una de estas entidades tendrán consigo los métodos get y set necesarios para el correcto funcionamiento de las entidades, y así poder adquirir y añadir la información necesario en la base de datos.

Dado que ya se ha explicado con anterioridad el modelo de datos y la relación que tienen entre ellos, las clases se explicarán en orden alfabético por facilidad. Las clases que se han creado en el proyecto son:

- **Contenido:** En esta clase se almacenan los contenidos que tienen cada uno de los ejercicios. Los parámetros que contienen esta clase son

  -id: Es el identificador de la clase. El dato que tiene es de tipo String.

  -**nombre_contenido:** Es lo que se usará como referencia y el nombre que se le pondrá al contenido. El dato es de tipo String.

  -**tipo:** Es el tipo de archivo que se está usando, es decir, si es imagen, vídeo, libro, pagina web, entre otros. El dato es de tipo String.

  -descripcion: Es un breve resumen de lo que contiene el archivo. El dato es de tipo String

  -archivo: Es el contenido del archivo, es decir, el enlace de la pagina web, el archivo con su directorio de la imagen o vídeo y demás.

- **Cuentas_log:** Se usa para almacenar las diferentes cuentas de los usuarios, tanto los terapeutas como los pacientes, siendo distinguidos entre sí por la variable rol. Las variables que contiene son:

  -id: Es el identificador de esta clase el cual, su tipo de dato es String.

  -contrasena: Es la contraseña que debe de poner el usuario para poder entrar a la página web. El dato es de tipo String.

-email: Es el email que debe de usar el usuario para poder entrar. Este email está enlazado con el email del paciente o terapeuta y debe de ser el mismo en ambos casos para que no de ningún tipo de error. El dato es de tipo String.

-rol: Es el rango que tiene el usuario que quiere iniciar sesión. El dato es de tipo String pero solo tiene está el rol de paciete o terapeuta.

- **Datos_Imagen:** Es la clase que almacena y maneja los datos que vienen de la aplicación de Google de Cloud Vision API. Las variables que se han creado para ella son:

-id: Es la variable que se usa como identificador para esta clase y es de tipo String.

-alegria: Es una variable de tipo String que nos indica el nivel de alegría que se obtiene con la aplicación.

-tristeza: Almacena el nivel de tristeza que calcula la herramienta y el dato es de tipo String.

-ira: Es la variable que indica el nivel de ira que tiene el usuario y el dato es de tipo String.

-sorpresa: Es la variable de tipo String que nos indica la sorpresa que aparenta el paciente.

-fotosubexpuesta: En este se almacena el parámetro que indica si la fotografía está subexpuesta o no, es decir si hay suficiente luz en la imagen. Es de tipo String.

-fotoborrosa: Aquí se almacena un dato de tipo String que indica si la foto es borrosa o no.

-verosimilitudfoto: Es de tipo String y se usa para saber la veracidad de la foto.

-ojoizquierdox: Es un dato de tipo float que indica la posición en el eje X del ojo izquierdo.

-ojoizquierdoy: Es de tipo float e indica la posición en el eje Y del ojo izquierdo.

-ojoderechox: Indica la coordenada X del ojo derecho y es de tipo float.

-ojoderechoy: Aquí se almacena la coordenada Y del ojo derecho y es de tipo float.

-seguridad: Se usará para almacenar la seguridad de la foto y es de tipo float.

- **Datos_Sensores:** Esta es la clase que se usa para manejar los datos de los sensores que miden el estado fisiológico del paciente. En el caso de este proyecto se usará un programa externo para suministrar los valores. Las variables usadas son:

  -id: Es el identificador de la clase y el dato es del tipo String.

  -dni: Es el valor del DNI del paciente quien se está haciendo la prueba. El dato es de tipo Sring.

  -pulsaciones: Es un dato de tipo float que representa las pulsaciones del paciente en ese instante.

  -tension: Es el dato que representa la tensión del paciente durante la prueba y es de tipo float.

  -galvanico: Es de tipo float e indica la capacidad galvánica de la piel del sujeto que se está haciendo la prueba.

  -hora: El dato es de tipo Date e indica la fecha en la que se ha hecho la prueba con el siguiente formato: DD/MM/YYYY (DD=Día, MM=Mes, YYYY=Año).

  -timestamp: Es un dato de tipo long y representa la fecha contada en milisegundos desde el 1 de enero de 1970.

- **Datos_TensorFlow:** Es la clase que se usa para los datos que se obtienen del modelo de inteligencia artificial de TensorFlow. Con esta clase se cogen los datos del modelo según su variable que luego se almacenarán en listas. Las variables de la clase son:

  -id: Es el identificador de la clase y es de tipo String.

  -x: Representa la coordenada X de la marca que nos da el modelo como resultado y es de tipo String.

  -y: Representa la coordenada Y de la marca, y nos la da el modelo. Es un dato de tipo String.

-marca: Es el valor que identifica la marca de la máscara del modelo, es decir, cual de los 486 puntos de la máscara representa. Es un dato de tipo String.

-nfoto: Es de tipo String y representa el número de la foto que es de entre todas las que se han hecho durante la ejecución del ejercicio del paciente.

-idborrar: Es el identificador que se usa para borrar todos los puntos de la máscara. Esto se hace, porque, como se ha explicado antes, el modelo da demasiados datos, que a la largar, saturarán la base de datos. El dato de este identificador es de tipo String.

- **Ejercicio:** Es la clase que se utiliza para almacenar y manejar todos los ejercicios que diseña el terapeuta. Cada ejercicio puede contener uno o varios contenidos, por ese motivo, en esta clase se almacena también una lista de los contenidos que contiene cada ejercicio. Las variables y tipos de datos de las variables son:

  -id: Es la variable que identifica cada ejercicio y es de tipo String.

  -nombre_ejercicio: Es la variable que determina el nombre de cada ejercicio. Este nombre lo indica el terapeuta a su parecer. Es de tipo String.

  -descripcion: Es una breve descripción sobre el contenido del ejercicio y es de tipo String.

  -duracion: Es una variable de tipo int e indica la duración del ejercicio.

  -contenidos: Es del tipo ArrayList<String> y representa los contenidos que tienen cada ejercicio.

- **Farmaco:** Es la clase que almacena toda la información referente a los fármacos que el terapeuta mandará a sus respectivos pacientes. Es una clase global, es decir, que todos los terapeutas tienen acceso a esta base de datos. Además de la información de los fármacos de forma individual, cada uno de ellos tiene una lista de interacciones que sufren con otros fármacos. Las variables son:

  -id: Representa el identificador de la clase para cada uno de los fármacos y es de tipo String.

  -nombre_farmaco: Es el nombre de cada fármaco y es un dato de tipo String.

  -descripcion: Es de tipo String y es un resumen/descripción del fármaco.

-lista_interacciones: Es de tipo ArrayList<String> e indica el nombre de los fármacos con los que tiene ciertas interacciones, ya sean positivas o negativas.

- **Listas:** Esta clase representa las listas que son necesarias para ejecutar ciertas funciones. El contenido de estas listas son otras clases, por ejemplo: List<Ejercicio>, la cual es una lista que apunta a cada uno de los ejercicios con los respectivos valores de las variables de esa clase. Esta clase de Listas no se usa para almacenar datos en la base de datos, solo se usa para poder llamar una lista que contenga otras clases. Se ha hecho esta clase independiente, y no dentro de otras clases, por facilidad y mantener cierta coherencia y estructura. Al no almacenar nada de esta clase en la base de datos, no es necesario usar un identificador, al contrario que en el resto de clases. Las variables de esta clase son:

  -ejercicios: Referencia a la lista de la clase Ejercicios y es del tipo List<Ejercicio>.

  -contenidos: Hace referencia a la clase contenidos y es del tipo List<Contenido>.

  -farmacos: Es del tipo List<Farmaco> e indica la lista de fármacos.

  -terapias: Indica las terapias que se han creado y es un dato del tipo List<Terapia>.

  -datos: Representa la lista de los datos suministrados por los sensores y es del tipo List<Datos_Sensores>.

  -datosimagen: Es de tipo List<Datos_Imagen> y se usa para representar los datos que suministra la aplicación de Google.

  -datostensor: Se usa para representar los datos que vienen del modelo de TensorFlow y es del tipo List<Datos_TensorFlow>.

- **paciente:** Es la clase que hace referencia al paciente. En esta está toda la información más o menos personal referida al paciente, desde su nombre y DNI hasta su dirección. Se puede ampliar a poner muchos más campos personales sobre el paciente, pero en este trabajo se han hecho:

  -id: Es el identificador para la base de datos de la clase paciente. Es de tipo String.

  -nombre: Hace referencia al nombre del paciente y es de tipo String.

  -dni: Indica el DNI personal del paciente y es un dato de tipo String.

-ap1: Es de tipo String e indica el primer apellido del paciente.

-ap2: Es el segundo apellido del paciente y es un dato de tipo String.

-telefono: Es el número de teléfono del paciente y el dato de esta variable es de tipo String.

-email: Representa el correo electrónico del paciente y es el mismo que usa cada paciente para iniciar sesión. El dato es de tipo String.

-dirección: Indica la dirección de la vivienda asignada al paciente. Es un dato de tipo String.

- **Registros:** En esta clase se almacena todo lo referido a los registros de inicio de sesión a la página web. Es el mismo registro tanto para pacientes como para terapeutas, ambos diferenciados por su rol, y también se deja indicada la fecha en la que inician sesión. Esta fecha de inicio de sesión también se usará de cara a determinar quien ha iniciado sesión y así determinar que puede o no ver en la página. Las variables que pertenecen a esta clase son:

  -id: De cara a almacenar esta clase en la base de datos se usará esta variable de tipo String como identificador.

  -dni: Hace referencia al DNI del paciente o el terapeuta que se obtendrá de la clase paciente o terapueta gracias al correo electrónico que utilice el sujeto para iniciar sesión. Es de tipo String.

  -email: Es de tipo String y es el correo electrónico necesario para iniciar sesión en la interfaz gráfica.

  -rol: Es una variable de tipo String aunque solo tendrá dos posibles valores: paciente o terapeuta.

  -hora: Es un dato de tipo Date y hace referencia a la hora de inicio de sesión del sujeto.

  -timestamp: Es una variable long e indica el tiempo en milisegundos empezando a contar desde el 1 de enero de 1970 hasta el momento que inicie sesión el sujeto.

- **Sesion:** La clase sesión es en la que se almacenan y controla las sesiones de cada terapia para cada paciente. En esta clase se tendrá información como la fecha de la sesión, el paciente, los datos de los sensores e información de la propia sesión. Las variables usadas en esta clase son:

-id: Es la variable que se usa en la base de datos como identificador y es de tipo String.

-fecha: Es la fecha en la que se realiza la sesión y es de tipo Date.

-id_paciente_terapia: Hace referencia a un identificador que relaciona el paciente con la terapia. Esto es así porque puede haber varias terapias con el mismo nombre o identificador, pero solo una asignada a un paciente en concreto. La forma de este identificador se diseña en la función del endpoint. Es un dato de tipo String.

-id_sesion: Es el identificador para hacer referencia a cada sesión y es de tipo String.

-id_datos_sensores: Hace referencia al identificador que representa los datos de los sensores de la sesión que ha realizado el paciente y es de tipo String.

-dni_paciente: Es una variable de tipo String e indica el DNI del paciente.

- **terapeuta:** Es la clase que se usa para almacenar todo lo referente a los datos del terapeuta, desde datos personales hasta una lista con los pacientes que tiene. Las variables de la clase son:

  -id: Es un dato de tipo String que sirve para identificar cada terapeuta en la base de datos.

  -nombre: Hace referencia al nombre del terapeuta y es una variable de tipo String.

  -dni: Indica el DNI del terapeuta en cuestión y es de tipo String.

  -ap1: Representa el primer apellido del terapeuta y es un dato de tipo String.

  -ap2: Es de tipo String e indica el segundo apellido del terapeuta al que se está haciendo referencia.

  -email: Es el correo electrónico que el terapeuta debe de usar para poder acceder a la aplicación web que se desarrolla en el trabajo. Es de tipo String.

  -pacientes: Es la lista de pacientes que tiene el terapeuta en cuestión. El tipo de dato que se usa es similar a los que se usan en la clase Lista, pero en este caso lo que contiene la lista es la clase paciente: List<paciente>.

- **Terapia:** En esta clase se almacenan todas las terapias asignadas a sus respectivos pacientes junto con los ejercicios y los fármacos asociados a la terapia que debe de ir siguiendo el paciente. Las variables son:

  -id: Para la base de datos donde se almacena toda la información, esta variable será su identificador de tipo String.

  -id_paciente_terapia: Es una variable similar a la explicada en sesión. Es un identificador que hace referencia a un paciente y su respectiva terapia que debe de seguir. Es de tipo String.

  -nombre_terapia: Es el nombre que se le da a la terapia y es de tipo String.

  -dni_paciente: Variable de tipo String que indica el DNI del paciente.

  -nombre_ejercicio: Dato de tipo String que indica el nombre del ejercicio que se le asigna a esta terapia.

  -nombre_farmaco: Nombre del fármaco que debe de tomar el paciente mientras dure la terapia. Es de tipo String.

  -cantdiad_farmaco: String que indica la cantidad del fármaco que se debe de tomar, es decir, si es de 100 mg, 200 mg y demás.

  -cada_cuantas_horas: Indica cada cuantas horas se debe de tomar el fármaco el paciente y es de tipo String.

  -durante_cuantos_dias: Hace referencia a la duración de la terapia y es de tipo String.

  -paciente: Este dato es del tipo de la clase paciente que se ha diseñado con anterioridad e indica el paciente al que se le asigna dicha terapia.

4.3. Simulación de los datos de los sensores.

Al principio, se tenía como objetivo el uso de cuatro sensores que medían parámetros fisiológicos del cuerpo humano. Estos sensores eran un casco BCI, un pulsioxímetro, un electromiógrafo y un sensor que medía la respuesta galvánica. Debido al COVID-19 se produjo una falta de stock de la mayoría de productos, incluyendo estos tipos de sensores, dificultad en recibir y enviar paquetes, y dificultad en entrar en contacto con otras personas. Todo esto provocó cierta imposibilidad en el uso de los sensores que

en un principio se iban a usar y faltas de recursos y medidas para poder trabajar con los mismos.

Debido a estas dificultadas se ha considerado la simulación de los datos de los sensores en un entorno de programación. Gracias a esto se obtendrían los datos necesarios para poder desarrollar la herramienta de adquisición de datos, pero sin la necesidad de poner en riesgo la salud debida la situación y también sin tener la necesidad de paralizar todo el proyecto ya que, independientemente de la simulación de los datos, el programa diseñado para la llamada del servicio del endpoint con el que se suben los datos simulados sería igual en el caso de tener los datos reales sacados directamente de los sensores.

Para generar los datos de los sensores se ha optado por diseñar un programa en el lenguaje de programación de C++, lo que se ha hecho es directamente programar una función relativamente simple que virtualice los datos de los distintos sensores y subir los datos para que puedan ser adquiridos por el programa principal diseñado en Eclipse, y ya, gracias al programa de Eclipse poder almacenarlos en la base de datos y usarlos según fuese necesario.

La generación de datos se ha hecho mediante comandos de generación aleatorio de números acotándolos a valores más o menos coherentes que se deberían tener en la realidad. Estos rangos que se han usado son: entre 50 y 160 para las pulsaciones, entre 80 y 140 para la tensión, y entre 0 y 30 para el factor galvánico. Estos valores son modificables según el criterio que se quiera usar, pero para el desarrollo del trabajo no se buscaba encontrar patrones sino tener una colección de datos que se puedan subir de forma fácil e instantánea al programa principal.

En cuanto a la subida de los datos, en este trabajo se usa la herramienta de endpoint que se basa en diseñar funciones que luego se ejecutan mediante una URL. C++ no tiene una herramienta para trabajar con URL y CodeBlocks tampoco tiene una librería implícita para facilitar dicho trabajo, por ese motivo se implementó la librería: libcurl [11]. Gracias a esta librería podemos usar las funciones del lenguaje Curl y así poder hacer uso de las URL como herramienta de comunicación de datos.

En cuanto a la implementación de la biblioteca de libcurl en el entorno de programación de Code::Blocks, hay una gran falta de información o se encuentra desactualizada, por ese motivo se ha visto necesario explicar una serie de pasos a seguir para poder trabajar con esta biblioteca:

1. Añadir al proyecto que se ha creado el link de la ruta del archivo libcurl.dll.a que está dentro de la carpeta lib del .zip descargado de la página oficial de de [11] y el archivo libcurl.a el cual está en la misma ruta que el anterior archivo:

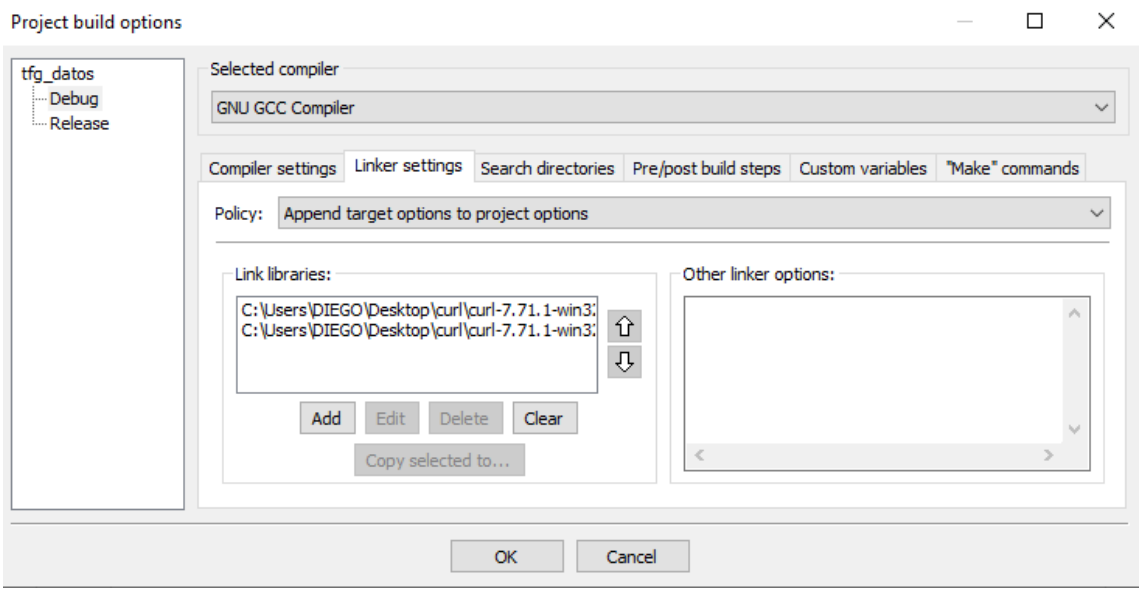

Ilustración 6Casillas a rellenar con la ruta de la librería

2. Se debe de completar para que la librería funcione correctamente es indicar en que ruta debe de buscar el programa para encontrar los archivos descargados de la librería libcurl. Esto se hace desde las opciones de Settings, Compiler, Search directories, y una vez allí, indicar en Compiler la ruta donde están todos los archivos del formato .h, y en Linker la ruta del directorio lib de la librería.

3. El último paso para que funcione correctamente es añadir el archivo de la librería: libcurl.dll en la carpeta donde está el archivo del formato .exe del programa que se ha creado en CodeBlocks, que está en la carpeta de Debug que está dentro de la carpeta de bin. En la misma carpeta en la que se acaba de añadir el archivo de la librería de libcurl se deben de añadir dos archivos más que se pueden encontrar fácilmente por internet los cuales son: libcrypto-1_1.dll y libssl-1_1.dll.

Una vez hecho todos estos pasos previos para poder conseguir que funcione la librería de libcurl correctamente, todos los pasos que se han ido haciendo para hacer que el programa de generación y subida de datos mediante comunicación URL funcione, junto con la explicación de algunos comandos de uso poco habitual son:

- Se incluyen todas las librerías necesarias para el desarrollo del programa: stdio.h, string.h, curl.h, STDIO.H, TIME.H, dos.h, time.h.

- Se abre el int main del código y seguido se inicializa el comando para los números aleatorios que dependan de la fecha en este instante con: srand(time(0)). Esto se hace para que cada vez que se ejecute el programa el generador de números aleatorios te de valores diferentes ya que cada vez dependerá de un valor de fecha diferente.

- Luego, se abre un bucle for en el que estará todo el resto del código. Este bucle es para subir tanta cantidad de datos como se quiera.

-A continuación, se deben de llamar tres primeras funciones necesarias para inicializar la librería:

```
CURL *curl;
CURLcode res;
curl = curl_easy_init();
```

-Y seguido, se duerme la función. Esto se hace porque en subir los datos se tarda un tiempo y así se evita que se superpongan los datos.

-Se abre una condición de si hay algo en curl, para comprobar que la librería funciona correctamente.

-A continuación se ha hecho la generación de los datos aleatorios para las pulsaciones, tensión y factor galvánico con el comando rand() seguido del símbolo "%" y el número hasta el que se quiere aleatorizar partiendo desde el 0. Seguido a esto se le ha sumado o restado el valor desde que parte y así obtenemos un valor aleatorio entre los rangos que nosotros deseamos y ese valor aleatorio siempre será diferente ya que depende de la fecha y hora en la que se ejecute el programa.

-Por último, se deben de ejecutar los comandos que subirán los datos mediante el protocolo de comunicación que nosotros necesitamos. Para ello, primero usamos un comando para indicar el tipo de mensaje que vamos a enviar, en este caso es de tipo POST, luego usamos un comando para indicar la URL a la cual queremos enviar la información de los sensores, seguido, otro comando que indica la dirección de la librería cuyo valor es siempre 1L, luego el protocolo que se usa, en este caso "https" y luego se ejecuta una función para asignar a una lista de strings con todo lo que queremos enviar. Después se indica el formato del contenido que se quiere enviar añadiéndolo en headers y otro comando para trazarlo con la lista de HTTP. Luego se debe de crear una cadena de caracteres y añadir en ella los datos que se le quieren pasar a esa dirección, que en este caso será el DNI del paciente quien está haciendo la prueba, el valor de las pulsaciones, el valor de la tensión y el valor del factor galvánico. A continuación, con un puntero se apunta a esa cadena de caracteres y con un comando de la librería libcurl se rellenan los campos del POST que estamos haciendo. Por último, se sale de la condición del if y con un comando de la librería se limpia todo lo que se ha hecho y comenzaría el bucle for si aún no ha terminado. Cuando este acabe, terminaría el programa al acto.

4.4. Google Cloud Vision API

Los recursos de la aplicación de Google se ejecutan durante el diseño de la interfaz gráfica de la aplicación, ya que no es necesario crear ningún endpoint para realizar el llamamiento de la aplicación. Por ese motivo se ejecutará el llamamiento de la aplicación directamente mediante código en lenguaje de JavaScript durante el diseño de la interfaz gráfica. Aún así, serán necesarios varios endpoints para el tratamiento de los datos, ver los resultados y borrarlos si fuese necesario.

Para poder ejecutar la aplicación se deben hacer unos pasos previos. Estos pasos previos consisten principalmente en conseguir unas claves necesarias para poder ejecutar la aplicación con los recursos que tiene Google en la nube. Para esto, se necesitará una cuenta de Google, habilitar la facturación en la misma y pedir unas credenciales las cuales te podrás descargar en local.

Una vez se obtienen esas credenciales se deberán ejecutar en el símbolo del sistema indicando la ruta en la que se han guardado las credenciales.

Si no ha resultado ningún error tras este proceso, en el símbolo del sistema se deberá de pedir un token con el siguiente comando:

```
gcloud auth application-default print-access-token
```

Y se deberá de generar un token. Este token es necesario para llamar a la aplicación de Google Cloud Vision API, ya que sin él no se tendrá permiso para ejecutar el programa, por lo consiguiente se nos dará un mensaje de error.

Este token que se obtiene dura un pequeño espacio de tiempo, de ahí que cada día que se quiera hacer una llamada al programa se requiera hacer este proceso para obtener un token diferente. También para obtener estas credenciales tras hacer la facturación se tienen solamente 3 meses de prueba, luego habrá que hacer un pago mensual o intentar tener las credenciales en orden como sea posible.

Para hacer la llamada a la aplicación de Google en JavaScript se debe de hacer una variable con una estructura como:

```
var settings = {
"url": "https://vision.googleapis.com/v1/images:annotate",
"method": "POST",
"timeout": 0,
"headers": {
"Authorization": "Bearer token",
"Content-Type": "application/json"
},
```

```
"data":
JSON.stringify({"requests":[{"image":{"content":base64},"featu
res":[{"maxResults":10,"type":"FACE_DETECTION"}]}]}),
};

$.ajax(settings).done(function (response) {
console.log(response);
}
```

En el código se puede observar en url la dirección donde se ejecuta la aplicación de Google, el método de la llamada, que en este caso es de tipo POST, en la autorización hay que escribir el token que hemos conseguido anteriormente y ponerlo donde pone token, el formato es de tipo JSON y en ese JSON con el que le enviamos la información a la aplicación se debe de poner la aplicación a la que se quiere acceder, que para este proyecto es la de FACE_DETECTION y la imagen en formato base64. Por último se ejecuta la aplicación y obtenemos como resultado todo lo de la variable response.

Si se quiere hacer algo con los datos se deberá de hacer entre el comando de console.log(response); y el último corchete, ya que ahí tendremos la variable response con todo lo que se nos devuelve como resultado.

El formato en base64 para imágenes, es un formato que te define toda la imagen mediante un código alfanumérico y es el que usa la aplicación de Google, pero tiene un problema, dicho formato da como resultado una cadena de caracteres demasiado grande como para poder guardarla en la base de datos y utilizarla en la aplicación de Eclipse. Por ese motivo, no se puede almacenar la foto que se hace en ese instante y se debe de ejecutar la aplicación de Google hacer la fotografía. Y ya, a continuación, almacenar los resultados que necesitamos de la misma.

4.5. Modelo de TensorFlow

Al igual que con la aplicación de Google Cloud Vision API, el modelo se ejecuta directamente junto con el diseño de la interfaz gráfica. Es un modelo de inteligencia artificial que modeliza una máscara para el rostro en tiempo real y da como resultado las coordenadas de cada uno de los puntos de la máscara.

Se ha visto necesario la implementación de un segundo modelo como lo es este de TensorFlow, ya que es mejor frente a la aplicación de Google en ciertos apartados, entre ellos que es código libre y que tiene una mayor velocidad. Pero todo esto se estudiará más adelante, en el punto 5. Análisis de Resultados.

El modelo que se va a utilizar pertenece a la biblioteca de TensorFlow. Esta biblioteca funciona para el aprendizaje automático, es decir, está especializada para IA tanto para el entrenamiento como para el desarrollo.

Concretamente, el modelo que se va a utilizar [12] es un modelo de inteligencia artificial con neuronas de tipo convolucional y una arquitectura de red neuronal

convolucional. Las neuronas de tipo convolucional se centran en captar características en concreto y darle un valor de importancia. Cuando este valor es más alto representa que esas características son más dominantes y así la red aprende a filtrar la información. Esto, cuando pasa a ser una red neuronal convolucional, las características que puede extraer de la imagen se vuelven más complejas y abstractas.

Para ejecutar el modelo con el que se va a trabajar, se deben de ejecutar diferentes funciones de forma consecutiva o paralela.

Además de las propias funciones que requiere el modelo para trabajar, se han diseñado otras funciones para hacer fotografías de momentos exactos. Ya que el modelo usado es un modelo de inteligencia artificial que predice donde estarán las máscaras del rostro, este modelo solo trabaja en base a un vídeo, al contrario que en la aplicación de Google que tan solo era necesario una imagen. Para el desarrollo del proyecto no es necesario un vídeo de la máscara ya que se quiere comprobar de forma pausada los movimientos del rostro de la cara y también se quiere aprovechar el almacén de la base de datos para guardar los distintos resultados y comprobarlos detenidamente. Este último imposibilita el guardar el vídeo de la máscara, ya que para usar los endpoints para guardar los datos en la base de datos y tratarlos se requiere cierto tiempo.

La implementación y acondicionamiento del modelo es:

-Se declaran dos variables globales para controlar el tiempo entre cada fotografía y el tiempo total del ejercicio.

-Se añade una función que coge como valores la región en la que se trabaja y los puntos de la máscara. Con esta función se igualan todos y cada uno de los puntos que nos da el modelo en la región de trabajo en 2D para comprobar que los valores son correctos y si es así, muestra por pantalla la máscara superpuesta con el rostro de la persona a la que se está grabando, siendo el rostro de la persona y junto con la zona donde se ve lo que graba la cámara toda la región del proceso de dibujo.

-A continuación, se inicializan más variables globales como son el modelo, los valores de las coordenadas, el ancho del video, el largo del video, el video, la imagen(canvas) y dos variables de dispersión. También se introduce una constante que el tamaño total del video y el estado de la configuración del modelo. Por último, se comprueba si la nube de puntos de renderizado está activa y en caso correcto, pone como verdadero el estado de la configuración del modelo.

-Luego, se programa una función que trabajará de forma asíncrona sobre la configuración de la cámara. En esta función cogeremos todos los valores del elemento video que habrá en pantalla, es decir, lo que la webcam estará grabando en todo momento. En el caso de que se haya modificado la configuración del tamaño del video deberá redefinirse aquí también.

-La siguiente función que se ha diseñado es la que se encarga de dar los resultados del modelo, es decir, donde se calcula la predicción de los puntos de

la máscara. Primero, le asignará a una variable los valores estimados del modelo, y a otra variable se le asignarán los valores de la imagen del video. A continuación, con una condición se comprobarán si hay valores en la predicción. Este valor deberá ser siempre verdadero, en caso de que sea falso se deberá a algún tipo de error. Una vez dentro de la condición se añadirán a una variable las predicciones que ha hecho el modelo y luego se llamará a una función que se explicará detenidamente más adelante y que realiza el proceso de llamar a un endpoint para almacenar en la base de datos los resultados obtenidos por las predicciones. Después, se abre un nuevo condicionante que comprueba una de las configuraciones que hemos hecho en el modelo, esta es para triangular la posición de cada uno de los puntos de la máscara. En caso contrario de no necesitar la triangulación de los puntos, se guardarán cada uno de los puntos de la máscara que nos da el modelo con sus coordenadas X e Y, y se guardarán en la variable ctx nombrada anteriormente, la cual se encarga de dibujar los puntos en la interfaz gráfica. Y, por último, se diseñan una serie de condiciones y bucles que nos facilita el modelo que son para cumplir las condiciones de la configuración que se le ha asignado, para diseñar el mapa con los puntos de las predicciones, adquirir datos de la base de datos que tiene el modelo de inteligencia artificial y para comprobar que todo se ejecuta correctamente.

-La función diseñada a continuación, es la primera que se tiene que ejecutar de todas las funciones del modelo. A esta se le llama cuando el paciente pulsa el botón para empezar con el ejercicio. Al principio de la función se inicializan al valor 0 las dos variables globales que se han creado antes para manejar el tiempo del ejercicio. Luego se llama a la función asíncrona que enciende la cámara y pone su configuración. A continuación, se le asignan al video, la imagen y a la variable ctx sus valores necesarios para trabajar, como tamaño, ancho, largo, y en el caso de la variable ctx también el color de la máscara y la escala. Seguido a esto, se guarda en una variable el modelo cargado que se usa para hacer las predicciones de los puntos del rostro. Y, por último, se llama a la función explicada en el punto anterior, la que se encarga de dar los resultados del diseño y se hacen comprobaciones sobre el video y las trazas de la imagen.

-La última función que se ejecuta durante el proceso de modelo de TensorFlow es la necesaria para el control del tiempo, la realización de las capturas en el momento exacto y envía los datos de las dos coordenadas, X e Y, de los 486 puntos de la máscara, mediante un endpoint que posteriormente se almacenarán en la base de datos. Para la realización, primero se abre un condicionante que comprueba si una variable global es inferior a cierto valor. En el caso de que no se cumpla, no se haría nada en esta función. En caso de que se cumpla, se entraría en el condicionante y se abriría otro condicionante que comprueba la otra variable del tiempo, el cual debe ser distinto a cierto valor. En el caso de que sí sea diferente sumaría un valor unitario y saldría de la función así hasta que el valor sea igual al que buscamos. Cuando ese valor sea igual al que buscamos se ejecutaría un bucle que se repetiría tantos puntos tenga la

máscara y añadiría las coordenadas de la máscara a las variables X e Y. Con estas coordenadas, el número del punto, y el número de la foto, el cual equivale a la primera de las variables de tiempo, se llamaría a uno de los endpoint que se encarga de guardas los valores de las coordenadas en la base de datos. Para conocer las fotos solo se necesitan las variables de tiempo, ya que la primera de ellas, la que hace que el programa se ejecute tantas veces se quiera se ha supuesto que será del mismo valor que el número de fotos que se harán. Para la separación entre captura y captura es necesario una sola variable que depende de una suma de un bucle ya que algunas de las funciones necesitan ejecutarse por medio de llamadas a endpoints y eso lleva cierta cantidad de tiempo, entre 2 hasta 10 segundos dependiendo del endpoint al que se esté llamando, y por ese motivo, con un simple bucle pasa el suficiente tiempo entre fotografías. Por lo dicho, querer hacer capturas en tiempos exactos se hace muy complicado dado que las llamadas de funciones por medio de URL no tienen una coherencia estable en cuanto al tiempo que duran, el mismo endpoint con los mismos datos puede durar unas veces un tiempo y otras veces otro, en función del servidor donde se ejecute y la velocidad de internet.

4.6. Funciones de los endpoints

Son el eje central del proyecto. Es un archivo del formato: ".java" en el que están incluido el endpoint con todos sus métodos que proporcionan diferentes servicios, es decir, todas las funciones que se ejecutan en el desarrollo del proyecto. Estas funciones se encargan tanto de almacenar, coger y borrar datos, y de ejecutar programas dependiendo de las entradas que se les suministre.

Estas funciones envían y reciben la información mediante mensajes de tipo POST, y para crear cada una de ellas se debe de tener en cuenta un nombre propio para la función, una ruta ("path") del mensaje POST que llama a la función, y las entradas, si las requiere.

Las operaciones que se desarrollan más adelante son las que están en endpoints dentro de la ilustración 2, y son necesarios para llamar a los distintos servicios o almacenar información en la base de datos. A su vez, cada uno de los métodos está vinculado al modelo de datos ya que se hace una llamada a la base de datos, ya sea para almacenar, extraer o modificar información.

En cuanto a la creación del archivo completo, primero se debe de cargar el paquete en el que se encuentra toda la información del proyecto y a continuación importar las librerías necesarias para poder trabajar. Estas librerías son las de Objectify, para poder acceder a la base de datos, luego una serie de librería para poder trabajar con los diferentes tipos de variables, como los son los vectores, las listas, las fechas y similares, y por último, las librerías necesarias para poder trabajar con los endpoints:

-com.google.api.server.spi.config.Api

-com.google.api.server.spi.config.ApiMethod

-com.google.api.server.spi.config.ApiNamespace

Una vez incluidas todas las librerías necesarias, se procede a crear la aplicación de los endpoints con una función tipo @Api. En ella se incluirá el nombre de la aplicación, la versión y el dominio en el que estará todo el trabajo.

Con todo esto hecho ya se puede proceder a abrir la función con el nombre que se le ha dado anteriormente y dentro de ella crear todos y cada uno de los programas que serán necesarios para el desarrollo del trabajo. La ruta necesaria para llamar a cada uno de estos programas se hace mediante en nombre de la misma función, pero añadiéndole el signo "_". Estos programas se explican a continuación ordenados de forma en la que se ha ido desarrollando a lo largo del proyecto:

- **insertar_terapeuta:** Esta función sirve para añadir un terapeuta a la base de datos. Hay que ir añadiendo de uno en uno, y para incluirlo es necesario insertar toda la información del propio terapeuta, es decir, como entrada se le debe de suministrar el DNI, el nombre, el primer apellido, el segundo apellido y un correo electrónico. Si se quisiese añadir más información, primero habría que añadirlo también en la clase terapueta.

  Dentro de la propia función, lo que se hace es buscar en la base de datos el DNI que se la ha suministrado como entrada en la base de datos, y mediante un condicional ver si está vacío o no. En caso de encontrar algo significa que ya está creado por lo que el programa termina indicando que el terapeuta ya estaba incluido. En el caso de que no exista, se crea una clase terapeuta, se incluyen todas las entradas, se almacena en la base de datos y termina el programa indicándote que el terapueta ha sido incluido.

- **cargar_terapeuta:** Con esta función, se obtiene la información almacenada en la base de datos de un terapeuta en concreto. Para trabajar con esta función es necesario suministrar como entrada el DNI. Con dicho DNI del terapeuta, mediante un filtro se buscará en la base de datos y con un condicional se determinará si existe o no el terapeuta.

  Una vez determinado si existe o no el terapeuta, en caso de que así sea se terminará el programa mostrando el terapeuta que se ha solicitad. Dicho terapeuta, pertenece a la clase con el mismo nombre y se mostrarán también así todos los parámetros del terapeuta pertenecientes a dicha clase. En el caso de que no se encuentre ningún terapeuta con ese DNI se mostrará un mensaje indicando que el terapeuta es erroneo.

- **borrar_terapeuta:** Gracias a una variable de entrada que indica el DNI del terapeuta que se pretende borrar, se buscará en la base de datos dicho terapeuta, al igual que en los casos anteriores. Pero en este caso, si no se encuentra ninguna coincidencia se indicará que el DNI es Nulo, pero en caso de encontrarse un

terapeuta, se borrará el terapeuta de la base de datos con el DNI que se ha indicado como entrada.

- **insertar_email_terapeuta:** Es un programa que te permite añadir un correo electrónico dentro de un terapeuta ya existente. Esto se ha desarrollado para poder cambiar de correo si el terapeuta lo requiriese o por si se olvidase añadirlo en la función de insertar_terapeuta. Esta función se puede hacer con el resto de variables que están dentro de la clase terapeuta, pero no se ha visto necesario.

  Para poder trabajar con esta función hay que suministrar como entrada tanto el DNI del terapeuta como el correo que se quiere añadir.

  Dicho programa, lo que hace es buscar si existe o no el terapeuta, gracias al DNI, y en caso negativo indica que el terapeuta no existe, y en caso positivo añade el correo electrónico al terapeuta.

- **insertar_paciente:** Similar al del terapeuta. Se dan las variables del DNI, nombre, primer apellido, segundo apellido, correo, teléfono y dirección. Se busca con el DNI en la base de datos, en la clase paciente si existe alguno con dicho DNI. En caso afirmativo, se indica que dicho paciente ya existe. En caso negativo, se inserta el paciente a la base de datos junto con todas las variables de entrada que se han indicado para inicializar el programa.

- **cargar_paciente:** Con ayuda del DNI del paciente que se da como entrada para esta función, se pretende buscar el paciente. Si no se encuentra ninguno, ya sea porque no existe o por error al escribir el DNI, se retorna un mensaje que lo indica. Pero en el caso de que dicho paciente sí que exista en la base de datos, se retornará como salida del programa el paciente de dicha clase con el DNI que se le ha indicado.

- **borrar_paciente:** Un programa que busca un paciente gracias a un valor de un DNI que se le indica como entrada. En el caso de existir dicho paciente se borrará de la base de datos indicando como salida un mensaje de borrado. Si dicho paciente no existe, se indica como salida que el valor del DNI es erróneo.

- **incluir_paciente_en_terapeuta:** Esta función no está implementada en la interfaz gráfica, es decir, que no es una función con la que se pueda hacer uso desde la interfaz que se explicará más adelante, pero sí que se ha hecho uso de ella. Lo que hace es insertar un paciente en un terapeuta, es decir, asignar a un paciente al terapeuta al que debe de ir. Este paciente se inserta dentro de la variable lista, con el nombre paciente que tiene cada uno de los terapeutas. Y, como cualquier lista, puede incluir en él más de un paciente, y todos ellos de la clase paciente.

  A esta función, se le dan como entrada dos variables, el DNI del paciente y el del terapeuta. Y con estos DNI se pretenden confirmar tres cosas, que el terapeuta con dicho DNI está en la base de datos, en caso negativo se indicará un mensaje de error indicando que el terapeuta no existe. Comprobar si el paciente ha sido

creado, si no, se indicará que no está creado. Y, por último, comprobar si el paciente ya estaba incluido. En el caso de que así fuera, esto queda reflejado con un mensaje de salida, pero, si este no está incluido, se incluirá el paciente en la lista del terapeuta y se devolverá un mensaje que lo indique.

- **lista_pacientes:** Este programa no necesita ninguna variable de entrada y lo que hace es retornar como salida la lista de pacientes que hay en la base de datos dentro de la clase de pacientes.

- **eliminar_paciente_de_lista:** Similar que en el programa de incluir_paciente_en_terapeuta, con este programa lo que se pretende es eliminar un paciente que esté dentro de un terapeuta. Para ello, se da como entrada tanto el DNI del paciente como del terapeuta, y se busca lo mismo que en el programa anteriormente indicado, que esté el terapeuta, que esté el paciente y que esté el paciente en el terapeuta. Si se cumplen las tres condiciones se eliminará el paciente con el DNI indicado. Y en el caso de que no esté, se mostrará un mensaje diciendo en cual de las tres partes se ha cometido un error.

- **incluir_contenido:** Una función para buscar incluir una variable de tipo contenido a la base de datos. Como entrada se le da el nombre del contenido, el tipo de contenido que es, una descripción del contenido y el archivo del contenido. Y lo que hace principalmente es buscar si el contenido con dicho nombre ya existe, y en caso negativo añade el contenido con las características que se le dan como entrada.

- **ver_contenido:** Con esta función se pretende tener como salida un contenido en concreto dentro de todos los posibles que se tienen en la base de datos. Para encontrar dicho contenido, como entrada a la función se le da el nombre del contenido y la función lo busca dentro de la base de datos. Si está lo muestra como salida y si no está, como salida muestra que el nombre del contenido ha sido erróneo.

- **borrar_contenido:** En el caso de que se de alguna equivocación y se quiera eliminar algún contenido en concreto se usa este endpoint. A el se le suministra como entrada el nombre de contenido que se pretende borrar de la base de datos y mediante una función con la librería de objectify se busca. Si no está sale un mensaje de nombre de contenido erróneo, y si está borra el contenido y sale un mensaje de contenido borrado.

- **cambiar_contenido:** Y este programa se usa en el caso de que se quiera cambiar el contenido manteniendo el nombre original. Para ello se introduce el nombre del contenido, el tipo de contenido que es, una descripción del contenido y el archivo del contenido. Y en este caso, al contrario que incluir_contenido, se pretende encontrar en la base de datos un contenido con el nombre indicado, en caso contrario, se mostrará un mensaje de nombre erróneo, y si sí existe dicho nombre de contenido, se borra el contenido de la base de datos y se añade uno nuevo con toda la información suministrada a la entrada.

- **lista_contenidos_total:** Para cuando se quieran obtener todos los contenidos almacenados en la base de datos, se usará este programa. Con este no es necesario introducir ninguna entrada y lo que hace es mostrar por la salida una lista completa de todas las variables de la clase contenidos que se tienen almacenadas en la base de datos.

- **crear_ejercicio:** Es el programa con el que se pretende insertar las variables de la clase ejercicio en la base de datos. Es necesario introducirle el nombre del ejercicio, la duración y la descripción del propio ejercicio.

  Como en el resto de funciones de este tipo, primero se busca en la base de datos si existe este ejercicio filtrándolo por el nombre del ejercicio. Sino existe se incluye en la base de datos junto con el resto de parámetros en la clase de ejercicio, y si ya existe, salta un mensaje indicándolo.

- **incluir_duracion_en_ejercicio:** Esta función se introdujo por si fuese necesario modificar el tiempo del ejercicio una vez introducido este. Para la función es necesario indicar el nombre del ejercicio y la duración del mismo. Este ejercicio se buscará, y se introducirá la duración.

- **ver_ejercicio:** Para ver el ejercicio y/o poder usar los datos del mismo usamos este programa. Primero, introducimos el nombre del ejercicio y se busca para ver si este está en la base de datos, si así es se saca como salida, sino, se indica que no se encuentra en la base de datos.

- **borrar_ejercicio:** Para borrar una variable de la clase ejercicio que no se desee, hay que llamar a esta función, a la que primero se le indicará el nombre del ejercicio. Esta función mediante un comando buscará el ejercicio en la base de datos para comprobar si se encuentra en ella o no. Si se encuentra se borrará y se mostrará un mensaje de que ha sido borrada, y si no se encuentra se mostrará un mensaje como que se ha introducido un nombre de ejercicio erróneo o que directamente no existe en la base de datos.

- **incluir_contenido_en_ejercicio:** Cada ejercicio puede contener una serie de contenidos, y estos contenidos se introducen al ejercicio gracias a esta función. Para ello hay que indicar el nombre tanto del ejercicio como del contenido y se comprueban 3 condiciones. Primero que el ejercicio esté dentro de la base de datos, luego que el contenido esté dentro de la base de datos, y, por último, que el contenido no esté ya incluido dentro del ejercicio. Si alguna de estas tres condiciones no se cumple, no se incluirá el contenido y se mostrará un mensaje que lo indique. Si se cumplen todas las condiciones, se incluirá el contenido en el ejercicio y se indicará que esto se ha producido.

- **lista_contenidos:** Esta función se diferencia de la función con el nombre de lista_cotenidos_total en que esta última mostraba todos los contenidos de la base de datos, pero, la de lista_contenidos, solo muestra los contenidos que hay incluidos en un ejercicio. Para conseguir esto, como entrada se suministra el ejercicio del cual queremos saber los contenidos que hay. Con este nombre de

ejercicio, se busca en la base de datos si está incluido, y si lo está, de entre todos los parámetros que hay dentro de la clase de ejercicio, extraemos solamente la lista de contenidos que tiene dicho ejercicio y lo devolvemos como salida. Si el ejercicio no existe, se indicaría como que el nombre del ejercicio no es correcto.

- **eliminar_contenido_de_ejercicio:** Para eliminar de la lista uno de los contenidos que pertenecen a un ejercicio, se usa esta función. Se indica el nombre del ejercicio al que debería pertenecer y del que pretendemos eliminar, y el contenido que queremos eliminar de la lista. Buscamos el ejercicio en la base de datos, y en el caso de que esté, buscamos dentro de la lista de contenidos del ejercicio, el contenido que pretendemos borrar, para ver si se encuentra o no. Si se encuentra dentro de él, borramos la lista de contenidos del ejercicio e introducimos una nueva sin el contenido que pretendíamos borrar y lo guardamos todo de nuevo en la base de datos. Si alguna de las condiciones no se cumple, se sale de la función y se deja indicado.

- **lista_ejercicios:** Para esta función no es necesario introducir ninguna entrada. Y lo que hace cuando se le llama es retornar por la salida la lista de todos los ejercicios que hay en la base de datos almacenados.

- **incluir_farmaco:** Es la función que se usa para introducir un fármaco, junto con todos los parámetros que contiene dicha clase, dentro de la base de datos. Por ello, para la entrada se mete el nombre del fármaco y la descripción del mismo. Con el nombre, se busca en la base de datos, y si no se encuentra, se introduce en la base de datos.

- **ver_farmaco:** Para ver la información que hay dentro de una variable de clase fármaco, en la entrada se mete el nombre del fármaco, con un comando se busca dicho fármaco en la base de datos, y si se encuentra uno con el mismo nombre, se da como salida la información del mismo.

- **borrar_farmaco:** Para eliminar de la base de datos un fármaco en concreto se usa esta función. Se introduce el nombre del fármaco, se busca dentro de la base de datos para ver si ya se había incluido anteriormente, ya que, en el caso de que no saltará un mensaje indicando que el fármaco es erróneo, pero si sí está el fármaco, se borrará de la base de datos y se indicará que se ha borrado el mismo.

- **incluir_interaccion_entre_farmacos:** Como se sabe que algunos fármacos sufren interacciones entre ellos, normalmente negativas, se ha visto necesario incluir algo que remarque que hay una interacción entre ellos. Con esta función se pretende introducir en una lista de cada fármaco con el que se sufre interacción. Para ello, se introduce como entrada el nombre de dos fármacos. A continuación, se busca en la base de datos si ambos fármacos han sido introducidos previamente en la base de datos, con que uno de los dos fármacos no esté incluido, se saldrá de la función y se indicará cual ha sido escrito erróneamente o no estaba introducido. Una vez se haya comprobado esto, se buscará, si en la lista de interacciones ya estaba incluido previamente el otro fármaco, ya que en el caso de que sí lo esté, se

indicará que dicha interacción ya había sido introducida con anterioridad. Pero en el caso de que no se hayan introducido previamente y de que ambos fármacos existen. Se extraerá la lista de interacciones de cada uno de ellos, se introducirá el nuevo fármaco con el que sufren interacción, se añadirá de nuevo la lista a cada uno de los fármacos y se volverá a guardar la lista completa con la nueva interacción de cada uno de ellos en la base de datos.

- **lista_de_interacciones_entre_farmacos:** Con esta función, se devolverá como salida la lista de interacciones que sufre un fármaco. Para esto, se introduce el nombre del fármaco del que se quiere extraer la lista, se comprueba que está en la base de datos y en el caso de que sí, se coge la lista de interacciones y se muestra a la salida.

- **lista_farmacos_total:** Para esta función no es necesario meter nada como entrada. Al llamarla, se muestra la lista completa de todos los fármacos que hay en la base de datos, es decir, todo lo que ha sido guardado en la base de datos que pertenezca a la clase fármacos.

- **crear_terapia:** Las terapias son el conjunto de todo lo que tiene que hacer el paciente durante un periodo de tiempo, por esto, para crear una, se hace con esta función que requiere de un nombre de terapia, el paciente al que se le asigna la terapia, el nombre del ejercicio que tiene que hacer, el nombre del fármaco que se debe de tomar, la cantidad de fármaco que se tiene que tomar, durante cuantas horas y durante cuantos días, todo esto como entrada. Una vez se le han suministrado las entradas, el programa, mediante una variable intermedia que se usará como identificador que es la suma del nombre de la terapia más el DNI del paciente, se buscará en la base de datos para comprobar si ya había sido creada. En el caso de que no, se comprueba si existe el paciente, a continuación, si existe el fármaco, y por último, si existe el ejercicio. Si todo existe y la terapia no está repetida se creará con todas las entradas asignadas. Si algo no se cumple se remarcará en la salida y no se creará la terapia.

- **ver_terapia:** Para ver una terapia, es necesario el DNI del paciente y el nombre de la terapia de la cual se quiere ver toda la información. Con estas entradas y llamando a esta función, si la susodicha terapia existe, se mostrará a la salida de la función.

- **borrar_terapia:** Al igual que en la anterior función, se necesita tanto el DNI del paciente como el nombre de la terapia. La función, mediante comandos lo buscará en la base de datos, y si está, la borrará de la base de datos y se indicará en la salida que ha sido borrada. En el caso de que no exista, se indicará que las entradas han sido erróneas.

- **lista_terapias:** Este programa no necesita ninguna entrada. Al llamarlo, genera a la salida una lista de todas las terapias que hay guardadas en la base de datos.

- **programar_cita:** Para programar una cita, es decir, implementar una sesión entre paciente y terapeuta, se debe de indicar a la entrada el DNI del paciente que asistirá

a la cita, y la fecha de la misma. Con esto, se hará un identificador teniendo en cuenta tanto el DNI como la fecha que se usará para buscar en la base de datos, dentro de la clase sesión, si dicha cita ya ha sido hecha o no, para evitar duplicados. En el caso de que no, luego se comprobará si el paciente existe, y si sí existe se guardará en la base de datos la sesión, teniendo en cuenta el identificador, la fecha y el DNI del paciente.

- **aplicar_terapia:** Una vez se tiene una sesión creada, se le puede aplicar una terapia a esta misma, es decir, algo que tendrá que hacer el paciente durante el desarrollo de dicha sesión. Para ello, llamando esta función e introduciendo el DNI del paciente, la fecha de la sesión, y el nombre de la terapia como entradas, primero se creará el identificador del DNI y la fecha para buscar en la base de datos la sesión y comprobar que existe. Si así es, se creará otro identificador que relaciona el paciente y el nombre de la terapia para buscarlo en la base de datos de la clase terapia. Si existe dicha terapia, se insertará en la sesión y se guardará en la base de datos. Si no saldrán mensajes indicando en que paso de los que hay que seguir no se ha encontrado en la base de datos lo necesario para aplicar la terapia en la sesión.

- **ver_sesion:** Para ver una sesión en concreto, es necesario, tras llamar este endpoint, introducir el DNI del paciente y la fecha a la que debería ir a esa sesión. Luego se busca en la base de datos si la sesión ha sido creada gracias a la variable que relaciona el DNI y la fecha, y si se encuentra una coincidencia en la base de datos, se indica a la salida.

- **cancelar_sesion:** Con esta función, se hace como con las funciones de borrar algo en concreto en la clase requerida. En este caso es necesario indicar el DNI y la fecha, y si se encuentra una sesión con esas características, se borrará la sesión de la base de datos y se indicará que se ha borrado. Si no se encuentra ninguna coincidencia se indicará que alguno de los parámetros de entrada es incorrectos.

- **insertar_datos:** Con esta función se añaden los valores de las pulsaciones, tensión y de la medida galvánica del paciente a la base de datos. Esta es la función que llama el programa hecho con Code::Blocks explicado anteriormente[0]. El programa realizado con Code::Blocks le suministra a esta función como entrada el DNI del paciente, y los valores de la tensión, de las pulsaciones y el coeficiente galvánico. Luego, dentro de la función se coge la fecha en formato long en milisegundos que han pasado desde el año 1970 hasta el momento en el que se le llama a la función. Esta variable tipo long la transformamos en una tipo Date, que ya indica el día, mes, año, hora, minuto y segundo en el que nos encontramos.

  Con esos dos formatos de la fecha, y junto con todos los valores de entrada, lo introducimos en una variable de la clase Datos_Sensores y se almacena en la base de datos.

  Para finalizar, se muestra por salida toda la información que se ha almacenado en la base de datos.

- **lista_datos:** En la interfaz gráfica, se ha visto necesario representar gráficamente los parámetros que se han medido en el paciente a lo largo de todas las sesiones. Para ello, llamando esta función e indicando como entrada el DNI del paciente del cual se quieren observar la información, con esta función se genera una lista de toda la información almacenada en la base de datos que sean de la clase Datos_Sensores para el paciente el cual se ha indicado el DNI al principio.

- **borrar_datos:** Si se desea borrar todos los datos de un paciente, si se llama a esta función, y se indica a la entrada el DNI del paciente, se buscará la lista del paciente de los datos de los sensores, y con un bucle que se repita tantas veces el tamaño de la lista, dentro de dicho bucle se irán borrando uno a uno toda la información almacenada en la base de datos de este paciente.

- **lista_datos_acotada:** Si no se quiere ver todos los datos a lo largo del tiempo de un paciente, sino que se quiere ver la información entre unas fechas en concreto, se llamará a esta función. Para esta función, se le indica como entrada el DNI del paciente del que se quieren ver los datos, la fecha mínima y la fecha máxima entre las observaciones que queremos hacer. Esas fechas, que están en formato Date, se pasan a formato long, es decir, contar los milisegundos desde el 1970 hasta la susodicha fecha. Luego, con el DNI, se busca en la base de datos de la clase en la que se almacenan los datos de los sensores, toda la información del paciente con el DNI suministrado como entrada y se genera una lista. Y, a continuación, dentro de un bucle del tamaño de la lista completa, con un condicional se compara si la fecha en formato long de cada uno de los componentes de la lista está dentro de las fechas límites indicadas. Si es así, no se hace nada, pero si están fuera de los límites se borra de la lista. Por último, esta lista en la que se han borrado los datos de los sensores que estaban fuera de fecha se devuelve como salida de la función.

- **borrar_datos_concreto:** Es una función con la que, si se le llama y se le indica un DNI de algún paciente, indicándole una posición en concreta dentro de la lista almacenada en la base de datos de los datos de los sensores del paciente, se borrará dicha posición en concreta, y devolverá el mensaje de datos borrados.

- **inserta_datos_vision:** Para introducir los datos que nos proporciona la aplicación de Google Cloud Vision API, en la base de datos que se usa a lo largo del proyecto, se utiliza esta función, a la cual hay que introducirle como entrada un identificador creado durante la interfaz gráfica, el valor de la alegría, el de tristeza, ira, sorpresa, el nivel de foto subexpuesta, de foto borrosa, de verosimilitud y los valores de las posiciones de los ojos, es decir, la posición en el eje x del ojo izquierdo, la posición en el eje y del ojo izquierdo, la posición en el eje x del ojo derecho y la del eje y del ojo derecho, y por último, la seguridad que se tiene de todo lo obtenido. Dentro de la propia función, todas estas variables de entrada introducidas, se insertan en una variable de la clase Datos_Imagen y se guarda en la base de datos.

- **borrar_datos_vision:** Para cuando se requiera de borrar todos los datos de la base de datos de la clase Datos_Imagen, se recurre a esta función a la que no es necesaria meter ninguna información como entrada. La función generará una lista

de todos los datos almacenados para conocer la cantidad de datos que hay en la base de datos. Con esto, se hace un bucle con el tamaño de la lista y se van borrado de uno en uno toda la información que estaba almacenada en la base de datos y se reporta como salida que los datos han sido borrados.

- **lista_datos_vision:** Para ver la lista de datos completa no se requiere de introducir ninguna entrada. Al llamar esta función, se generará una lista de la información almacenada en la base de datos sobre los valores que tenemos en la clase Datos_Imagen y se mostrará a la salida.

- **resultados_vision:** Es el endpoint de mayor tamaño de entre todos los que se han desarrollado, en cuanto a cantidad de líneas de código. Lo que se pretende hacer en esta función es evaluar los resultados que proporciona la aplicación de Google Cloud Vision API. Se tiene que ejecutar cuando ya se hayan almacenado los datos de la aplicación en la base de datos y no es necesario introducir ningún tipo de entrada.

  En cuanto al desarrollo de la función. Primero se genera una lista de los datos almacenados en la base de datos de la clase Datos_Imagen. Con esta lista, se observa la cantidad de capturas que se ha hecho, es decir, el tamaño de la lista y se transforma a una variable de tipo String. Luego se crean una serie de variables y se inicializan a 0. Estas variables indican si el paciente se mueve lentamente, de forma moderada o mucho, la seguridad que se tiene, y una cuenta de las veces que se le ha observado al paciente alegre, enfadado, triste o sorprendido. Todas estas variables son enteros de valor igual a 0 excepto la seguridad, que también tiene valor 0 pero es de tipo float.

  Una vez se han inicializado todas las variables comienza un bucle de tamaño la lista que estaba almacenada en la base de datos y primero, se va sumando el valor de seguridad que tiene cada dato de la lista. Luego, se pasa a comperar lo que se ha movido el sujeto cuando se está al menos en la segunda posición de la lista.

  Para comparar lo que se ha ido moviendo el paciente se han escogido unos parámetros de las posiciones arbitrarios y se han ido comparando entre ellos, si están dentro de esos parámetros, se suma de uno en uno a la cuenta de ese parámetro y así se puede comprobar que es lo que más se ha hecho durante el desarrollo del ejercicio, si moverse levemente, de forma moderada o mucho.

  A continuación, se ha ido comprobando las veces que ha estado enfadado, alegre, triste o sorprendido haciendo que un contador sume de uno en uno cada vez que la aplicación nos devuelva un parámetro diferente a “UNLIKELY”.

  Con esto volvería a empezar el bucle. Y cuando termine el bucle, se pasaría a la muestra de los resultados. Primero, a base de condicionales, se compara el número de veces que se ha movido de forma leve, moderada o mucho, con respecto el total de la lista. Si está por encima del 50% saldrá un mensaje que indicará que se ha estado moviendo de esa forma la mayor parte del tiempo. Si el contador el porcentaje es 0 se dirá que no se ha movido de esa forma durante todo el ejercicio.

Si está entre el 35 y 50% se indicará que se ha movido casi la mitad del tiempo de esa forma. Y si está entre el 0 y el 35% se reflejará que se ha movido de esa manera durante muy poco tiempo.

Luego, se calcula la seguridad media que se tiene con todas las imágenes dividiendo el parámetro de seguridad entre el tamaño de la lista y multiplicándolo por 100.

Por último, con la cuenta de veces que ha mostrado cada uno de los sentimientos, a base de condicionales, si no se ha mostrado ninguna vez dicho sentimiento se dirá que no lo ha mostrado ninguna vez. Si lo ha mostrado una vez se indicará que lo ha mostrado una vez. Y si lo ha mostrado más de una vez, se indicará la cantidad de veces que se ha mostrado.

Estos resultados se representan en su mayoría mediante una frase que representa el resultado del mismo. Todas estas frases que representan los resultados se van guardando en un vector, el cual será lo que se dará por salida de la función.

- **lista_datos_ejercicio:** Con esta función se comprueba tan solo los datos de los parámetros de los sensores almacenados en el instante de hacerlo. Para ello, dando como entrada el DNI del paciente del cual se quiere obtener la información. Se busca en la base de datos la información que hay almacenada para ese paciente, y con un límite de un día, dentro de un bucle del tamaño de la lista se compara, gracias al parámetro de la fecha en formato long, con la fecha en este mismo instante en formato long del día anterior y el día siguiente (se hace esto por poner unos límites máximos y mínimos, se puede modificar dichos límites). Si está dentro de los límites no se hace nada a la lista generada, y si está fuera, se elimina de la lista. Y se muestra por la salida la lista que se queda, con solo la información de los datos de los sensores de un solo día de margen.

- **insertar_cuenta:** Para entrar a la aplicación web, es decir, a la interfaz gráfica, primero, quien quiera entrar, debe de iniciar sesión. Esto se hace por varios motivos como lo son, diferenciar entre lo que ve el paciente y el terapeuta, y focalizar la información que puede ver cada uno dependiendo de quien haya iniciado sesión.

  Para insertar los usuarios se usa esta función. Primero introduciendo una contraseña, el correo del usuario y el rol al que pertenece. Después, se genera una variable intermedia que representará el identificador de la cuenta y que será la suma entre el correo y la contraseña. Luego, se buscará en la base de datos el correo para comprobar si ya había sido creado, y en el caso de que no, se añadirá a la base de datos dentro de la clase de Cuentas_log el usuario con las características suministradas como entrada.

  No se ha visto necesario la comprobación de si dicho correo existe ya que así se da la posibilidad de crear usuarios externos, como lo podría ser un invitado o el administrador.

- **ver_cuenta:** En esta función se introduce como entrada tanto la contraseña como el correo. Con estas variables se genera un identificador y luego comprobamos si

el correo está o no en la base de datos. Si no se encuentra se mostrará un mensaje de cuenta incorrecta. Pero en el caso de que sí se encuentre en la base de datos alguna cuenta se mostrará por la pantalla.

- **borrar_cuenta:** Si se quiere borrar alguna cuenta para cambiarla o porque se ha cometido por error se usa esta función. Cuando se le llama, primero hay que introducir tanto la contraseña como el correo. Luego, al igual que las dos funciones anteriores, se genera el identificador que contiene a ambas y se comprueba el correo en la base de datos. Si hay coincidencia, se borra de la base de datos y se indica con un mensaje. Si no, también se quedará indicado con un mensaje que no se ha encontrado la coincidencia.

- **incluir_registro:** Una vez que un paciente o terapeuta inicia sesión, dicho inicio se guarda en la base de datos como un registro gracias a esta función. Metiendo como entrada el correo electrónico de quien ha iniciado la sesión. En la función se coge el tiempo del sistema, es decir la hora en este instante y con el correo se busca la cuenta. Si la cuenta no coincide con ninguna almacenada en la base de datos salta un mensaje de error y si coincide se busca el rol asignado ha dicha cuenta de correo.

  Dependiendo de si es paciente o terapeuta se buscará en la base de datos dentro de la clase paciente o terapeuta buscando al que tenga un correo electrónico como el que se ha indicado como entrada, y dicho DNI se guarda también junto con el correo, la fecha tanto en formato Date como en long y el rol en la base de datos como una clase de tipo Registros.

- **ver_ultimo_registro:** Esta función busca al último que ha iniciado sesión y devuelve la información del mismo.

- **incluir_dato_tensorflow:** Cada una de las marcas que proporciona el modelo de TensorFlow se van almacenando en la base de datos de marca en marca y dependiendo también a su vez de la foto a la que se hace referencia. Con la llamada de esta función, al introducir la marca, el número de foto y las coordenadas tanto del eje x como del eje y se guarda toda esa información en la base de datos dentro de la clase de Datos_TensorFlow

- **lista_datos_tensorflow:** Es una función que no requiere ninguna entrada. Cuando se le llama genera una lista de todos los datos que están en la base de datos dentro de la clase Datos_TensorFlow y los muestra a la salida.

- **borrar_datos_tensorflow:** Al llamar esta función, lo que hace es borrar toda la información de la base de datos de la clase de Datos_TensorFlow. Esto lo hace borrando los datos de uno en uno con un bucle del tamaño de la lista completa de la clase Datos_TensorFlow.

- **ver_datos_tensorflow_individual:** Por último, con esta función lo que se pretende es, ver unas coordenadas en concreto de la marca y foto que queramos ver y esté previamente almacenado en la base de datos. Para ello, se crea una lista de toda

información que haya en la base de datos en la clase Datos_TensorFlow filtrando por la marca que queramos ver. Y, para acotar a la foto que deseamos, devolvemos como salida la información que esté en la lista que pertenezca a la posición de la foto que se desea ver.

Una vez termina el desarrollo de la parte del backend, se comienza con el desarrollo del frontend como lo sería la parte del diseño de la interfaz gráfica.

4.7. Interfaz gráfica

La interfaz gráfica pertenece al frontend dentro de la arquitectura global del diseño del sistema como se indica en el punto 4. Diseño del Sistema. Esta interfaz gráfica es con lo que tiene contacto directo el usuario. Es una aplicación web con la que se puede interactuar con distintos botones, selecciones o entradas de texto, y con la que se visualiza la información que se almacena en la base de datos.

El diseño de la interfaz gráfica se ha hecho también dentro del entorno de programación de Eclipse en la carpeta de webapp. Se han desarrollado 3 ficheros diferentes, es decir, lo que equivaldría a 3 enlaces distintos.

El primero de ellos representa la página de inicio, al cual se le ha llamado index. Este es en el que se realiza el inicio de sesión y el que hace que derives a una página u a otra.

Los otros dos, se les ha puesto como nombres terapeuta y paciente. Ambos son muy similares. Las diferencias principales entre ellos es que el paciente tiene menos opciones entre las que elegir que el terapeuta y también solo puede acceder a su propia información, es decir, tiene los permisos limitados. Al ser tan similares entre ambos, primero se pasará a explicar todo lo que tiene en completo la página del terapeuta y más adelante, se explicará tan solo las diferencias que tiene con la página del paciente.

En cuanto a la programación de la interfaz gráfica, se programa el diseño de la página mediante HTML5 y las funciones desde el lenguaje de programación de JavaScript. Las tres páginas tienen el mismo formato. Primero se configura el contenido y se pone la referencia de distintas librerías necesarias. Luego se definen los estilos que se van a usar para el diseño de la página. A continuación, se empieza a programar el diseño de la página. Y, por último, se hace la programación de las diferentes funciones en JavaScript.

Para la estética de la página se ha usado la librería de Material.Design [7], la cual proporciona diseños de tarjetas, desplegables, botones iconos y elementos similares. También, para el diseño de las gráficas se ha usado Google Charts [8] con el que se pueden hacer diseños de todo tipo de gráficas suministrando los datos necesarios para ellos.

Para el diseño de la página, se ha necesitado usar diferentes tipos de plantillas e ir llamándose de unas a otras, es decir, que la página es dinámica. Para ello se ha usado la librería de jsviews [12]. Para trabajar con las plantillas, lo que se ha hecho es ir llamando a cada una de las plantillas del diseño de la página con un nombre, dentro de estas plantillas, si es necesario cambiar de diseño de página, primero se llama a una variable que se ejecuta al pulsar o activar una acción concreta. Esta variable llama a la función en la que se ejecutará todo el programa necesario como es llamar endpoints, hacer bucles, condicionales y demás. Por último, esta función llama a un enlace de la plantilla, en el cual se puede almacenar datos, función que se usará muy a menudo, y este enlace llevará a la plantilla del nuevo diseño de la página. Todo esto se explica más detalladamente dentro del .html de terapeuta, ya que es en el que más se desarrollan los pasos.

Los 3 .html se explican de forma ordenada, es decir de arriba hacia abajo, ya que, aunque se va saltando de una parte del código a otra es más entendible de forma ordenada.

Las tres diferentes páginas que se han diseñado son:

- **index:** Al principio, se hace la configuración del contenido y se insertan los enlaces de las diferentes librerías que se van a usar. Luego se añaden todos los estilos. Para esta página se va a usar el estilo de la tarjeta, el cual consiste en una tarjeta de tamaño 430px, otro estilo para el título, que lo que hace es poner el tamaño de letra a 20 px y otro estilo para poner las letras en negrita.

  En cuanto al diseño, no se tienen plantillas al contrario que en los otros dos .html y el diseño consiste en una tarjeta, con un título (Plataforma de Salud Emocional y Psicológica), dos campos a completar mediante texto con una descripción en estos que indica que es lo que se tiene que introducir en dichos campos (email y Contraseña). Y, por último, dentro también de la tarjeta, se tiene un botón en el que pone: Iniciar Sesión como se observa en la Ilustración 7. Al pulsar dicho botón se llama a la función login.

  Dentro de las funciones, solo se tiene la de login que se le llama con el botón antes indicado. Dicha función no requiere de ningún parámetro de entrada y lo que hace es leer lo que se ha escrito dentro de los campos de email y contraseña y os guarda cada uno en una variable. Luego, se llama al endpoint de “ver_cuenta” introduciendo tanto la contraseña como el email y se guarda la salida de dicho endpoint en una variable en la cual comprobamos el rol (paciente o terapeuta) y se compara la contraseña de la respuesta del endpoint con la contraseña que se ha escrito en la página web. Si ambas coinciden se enviará al .html del paciente o del terapeuta según el rol correspondiente. Si no coinciden aparecerá un mensaje de alerta indicando que la contraseña o el correo son incorrectos. Con esto terminaría la función y la página.

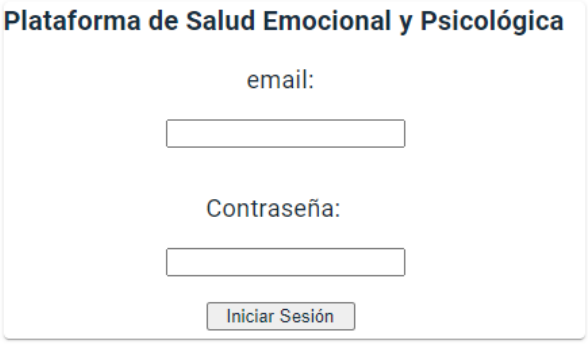

Ilustración 7 Diseño de la página de inicio

En la imagen superior se observa como quedaría el diseño de la página completo. En el que se puede observar los dos campos a completar junto con el título y el pulsador para la llamada de la función.

- **terapeuta:** Se accede a esta página tras iniciar sesión como terapeuta desde la página de inicio. En cuanto al programa que engloba toda la página, primero, se cargan las diferentes librerías, entre las que destacan las de jsviews, las del programa de TensorFlow y las de los diseños tanto de la página de Material Design, como las de Google Charts.

Luego, se empiezan a diseñar los diferentes estilos que se van a usar para el diseño de la página. Estos diseños engloban el tamaño y forma de las tarjetas, cuando una letra está en negrita o no, tamaños de letra y márgenes concretos a seguir como pueden ser para poner títulos o descripciones, formas y posiciones para las fotos que se usarán, y estilos para los distintos tipos de canvas que se usan a lo largo del diseño de la página.

Al terminar con los estilos, se cierra la parte de la cabeza del .html y se abre la parte del cuerpo. En este caso, tanto para la página del terapeuta como del paciente, se usa una estructura de diseño similar. Este diseño consiste en una cabecera, en la que se observa un título, en este caso: Plataforma de Salud Emocional y Psicológica. Luego, se tiene una barra de navegación desplegable en la que se pueden seleccionar distintas opciones, es decir, servicios a los que acceder. Estos servicios, tienen a ellos un icono representativo. Los servicios son: Pacientes, Terapias, Ejercicios, Contenidos, Farmacos y Cerrar sesión, llamando cada una de ellos al pulsarlos a los identificadores: “idpacientes”, “idterapias”, idejercicios”, “idcontenidos”, “idfarmacos”, “idlogout”, respectivamente. Y, la última parte de la estructura del diseño de la página es la plantilla. Es decir, depende de la plantilla que se seleccione habrá un contenido u otro. Estas plantillas se sitúan en la zona en blanco de la ilustración 8:

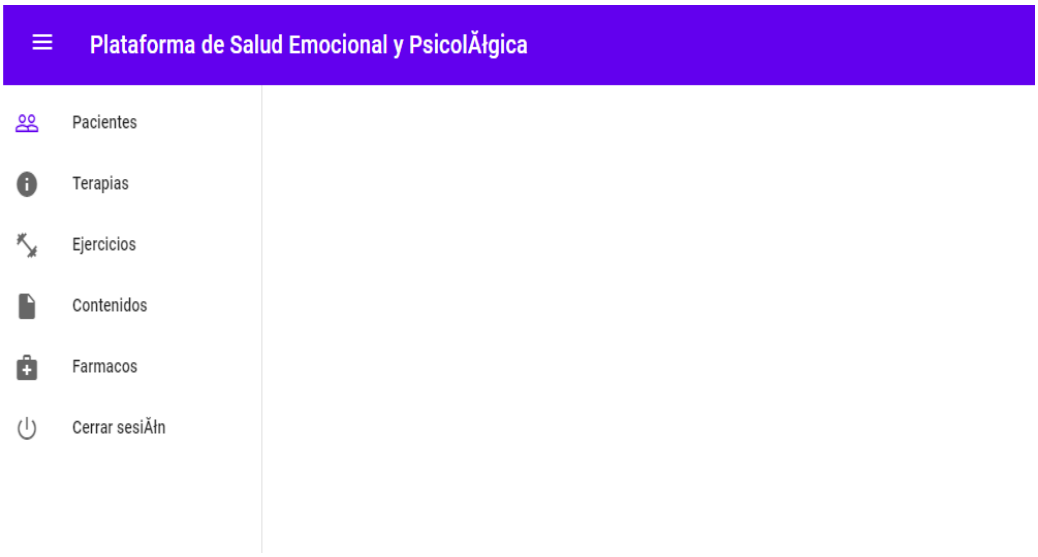

Ilustración 8 Pantalla principal sin escoger template

Tras la estructura principal, se ha diseñado la plantilla de "contenidoPacientes". Cuando se llama a esta plantilla, lo que se hace es un bucle añadiendo los pacientes con su nombre y el rol de paciente en una tarjeta. A esta tarjeta se le han añadido dos iconos abajo a la derecha. Estos iconos actúan como pulsadores que llevan al identificador de "cargar_unpaciente" y "cargar_graficos". A ambos se le suministra como entrada el DNI del paciente al cual se le ha pulsado en dicho icono. La plantilla se vería como la Ilustración 9:

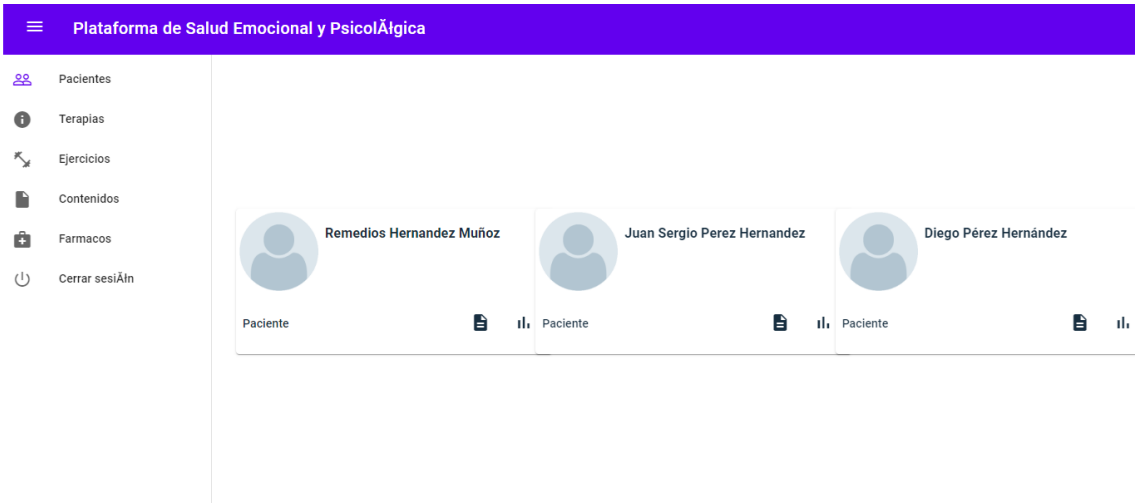

Ilustración 9 Plantilla que muestra a todos los pacientes

La siguiente plantilla que se ha creado es “contenidoUnPaciente”. En ella, aparece en formato tarjeta, toda la información personal de un solo paciente. Esta información es el correo electrónico, el DNI, el nombre, la dirección y el teléfono. También aparece la opción de, mediante un pulsador con el icono de un diagrama de barras, ir al identificador de “cargar_graficos” dando como entrada el DNI del paciente.

La plantilla se observa en la Ilustración 10:

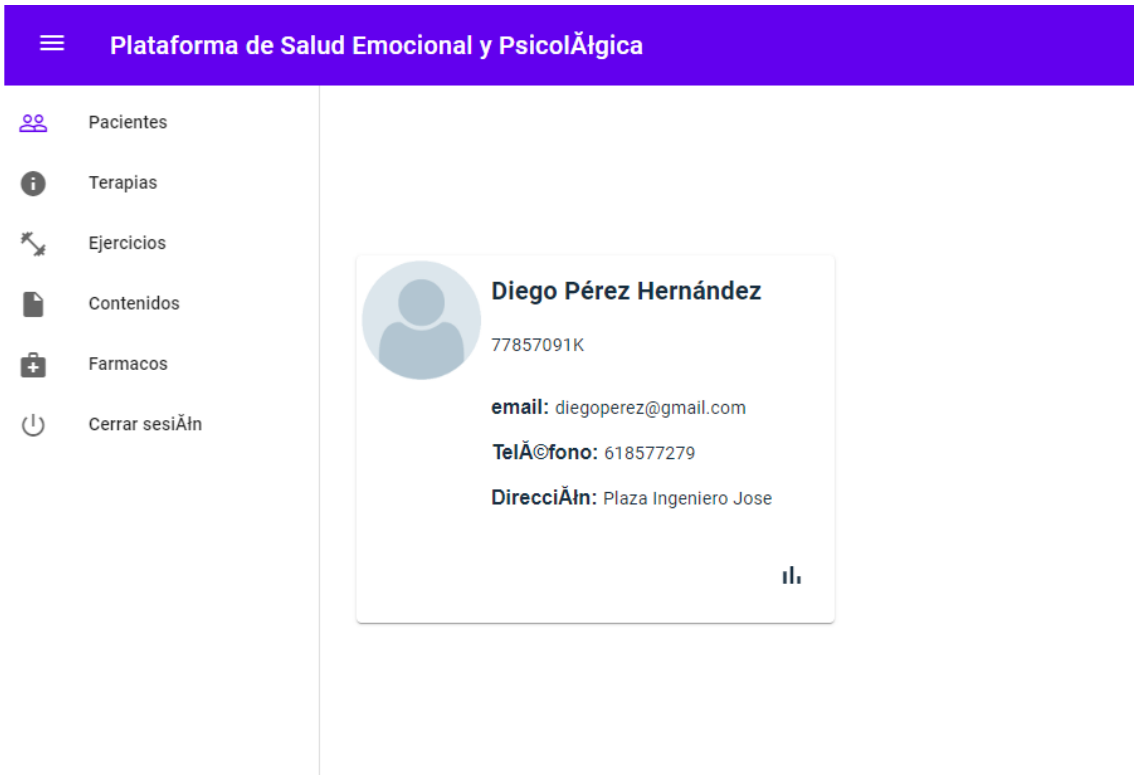

Ilustración 10 Plantilla del contenido de un solo paciente

La plantilla de “ContenidoEjercicios” se basa en dos textos que en realidad son dos pulsadores. En los que ponen ver lista de ejercicios, y crear ejercicio nuevo. El primero de estos lleva al identificador de “cargar_ejercicios” y el otro a “crear_ejercicio_pantalla”. Luce como la Ilustración 11:

Ilustración 11 Plantilla que te pide elegir entre ver la lista de ejercicios o crear nuevo ejercicio

Para plantilla de "contenidoCargarEjercicios" que se ve en la Ilustración 12:

Ilustración 12 Plantilla que muestra la lista de ejercicios creados

Se muestra la lista de ejercicios que se han creado. En cada uno de los ejercicios, se muestra tanto la duración de los mismo como los contenidos que tiene dicho ejercicio. También tiene como título el nombre del ejercicio y una breve descripción. Por último, se han añadido unos pulsadores, el primero de ellos lleva a "pantalla_incluir_ejercicios" con la entrada del nombre del ejercicio. El segundo a "eliminar_ejercicio" también con entrada necesaria del nombre del ejercicio. Y el tercero a "pantalla_elegir_tipo_ejercicio" con una entrada también del nombre del ejercicio.

La plantilla “contenidoelegirtipoejercicio”, Ilustración 13, es similar a la de “Contenidoejercicios”, pero en este caso se muestra como texto “Hacer ejercicio con medición TensorFlow” y “Hacer ejercicio con medición Cloud Vision Api” como se ve a continuación:

Ilustración 13 Plantilla para elegir el tipo de medición para hacer el ejercicio

Cada una de estas te lleva al identificado “tensor” y “pantalla_hacer_ejercicio”. Para el primero no se envía ninguna información adicional, pero para el segundo se incluye el nombre del ejercicio.

La siguiente plantilla, Ilustración 14, es como la anterior, vuelve a ser textos que funcionan como pulsadores y tiene el siguiente diseño:

Ilustración 14 Plantilla para elegir el tipo de contenido a visualizar

En esta plantilla, pulsando cada uno de los textos se puede elegir a ir al identificador de "cargar_contenidos_videos", "cargar_contenidos_imagen" o "cargar_contenidos_otros", pulsando "Ver lista de contenidos en formato vídeo", "Ver lista de contenidos en formato imagen" o "Ver lista de contenidos en otro formato".

La plantilla que se usa para ver los contenidos que se tienen en formato vídeo es como se observa en la Ilustración 15:

Ilustración 15 Plantilla para ver los contenidos en formato vídeo

Con esta plantilla, tantos contenidos se tengan con que sean de formato video se muestran aquí como se ve en la imagen superior. Cada contenido, tiene su nombre, una descripción y el video, en estos casos de youtube, para poder visualizarlos directamente desde esta página. Para obtener ese enlace de youtube y poder insertarlo aquí, se necesita usar el tipo "iframe" con una configuración concreta que nos proporciona youtube cuando se quiere compartir un vídeo con la opción de

insertar. También, desde esta misma página, se tiene un pulsado el cual lleva al identificador de “pantalla_crear_contenido_video”

En “contenidoPantallaCrearContenidoVideo” se tienen varias entradas de texto y un pulsador como se observa en la Ilustración 16:

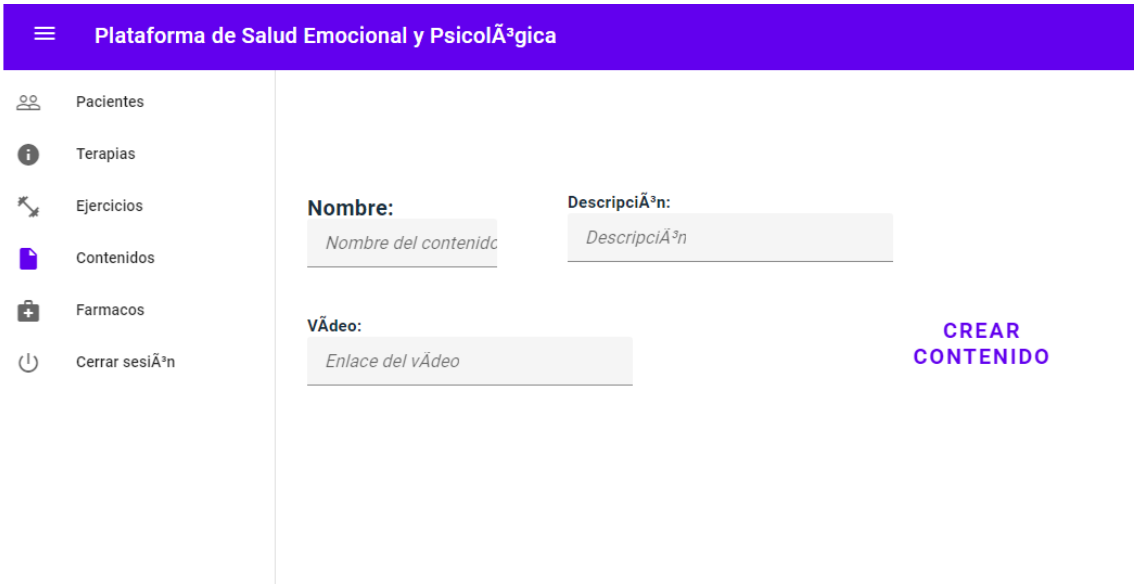

Ilustración 16 Plantilla para crear contenido en formato vídeo

Estas entradas para texto son para introducir el nombre, descripción y el enlace del vídeo. Tras pulsar el texto de “Crear Contenido” se irá al identificador de “crear_contenido_video” con el que más adelante se accederá a la información de los distintos parámetros.

Para visualizar los contenidos de las imágenes es de forma similar a los vídeos y se puede ver en la Ilustración 17:

Ilustración 17 Plantilla para ver los contenidos en formato imagen

Esta plantilla se llama “contenidoContenidosImagen” y con ella se ven la lista de las imágenes que se han guardado. Para hacer uso de estas imágenes, se deben de guardar dentro de la carpeta que se está usando para el diseño de las páginas. Además, como en el de los vídeos, este también tiene un pulsador que lleva a “pantalla_crear_contendio_imagen”.

También se pueden crear contenidos del formato imagen, Ilustración 18, con la plantilla “contenidoPantallaCrearContenidoImagen”:

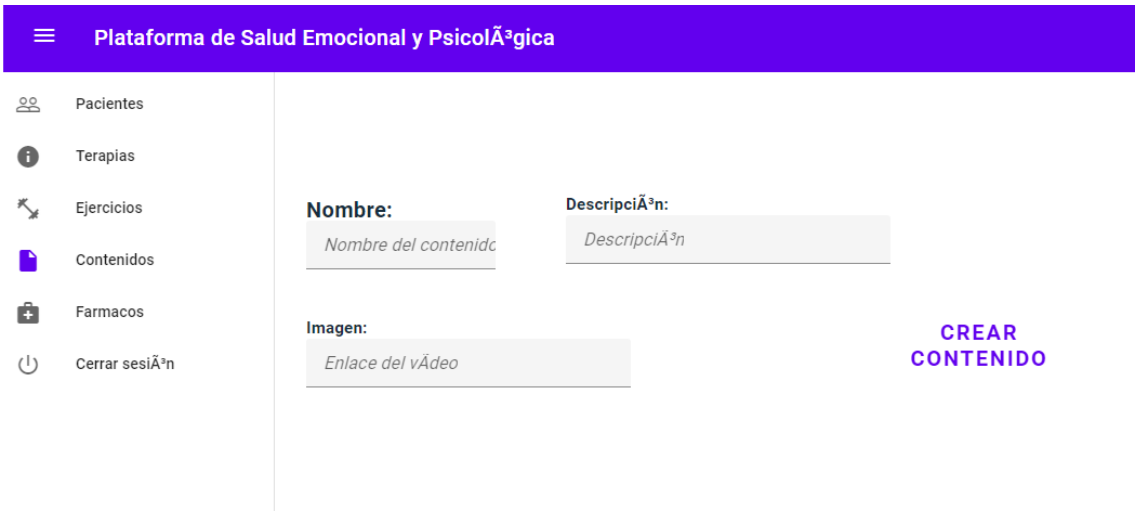

Ilustración 18 Plantilla para crear contenidos de tipo imagen

Muy similar con el que se crean los contenidos de formato vídeo, pero este, al pulsar en el texto de “Crear Contenido” lleva a “crear_contenido_imagen”.

El último tipo de los contenidos es el que contiene el resto de tipos de contenidos y la plantilla se llama “contenidoContenidosOtros” y está en la Ilustración 19:

Ilustración 19Plantilla para ver el resto de tipos de contenidos

Esta plantilla, muestra los contenidos de la base de datos que no son del tipo video o imagen. Se muestran en formato tarjeta con toda la información relevante con respecto el contenido. También tiene un pulsador que lleva a “pantalla_crear_contenido_otros”.

Para este tipo de contenidos también se tiene una plantilla con la que crear los contenidos, Ilustración 20:

Ilustración 20 Plantilla para crear los contenidos que no sean de formato vídeo o imagen

La plantilla se llama "contenidoPantallaCrearContenidoOtros" y tiene esta vez 4 entradas de texto, una más que el resto para indicar el tipo de contenido que es, y el pulsador lleva a "crear_contenido_otros".

En la plantilla "contenidoFarmacos" se muestran toda la lida de los fármacos almacenados en la base de datos, con la información que tienen. Esta información se ve en formato tarjeta en la Ilustración 21:

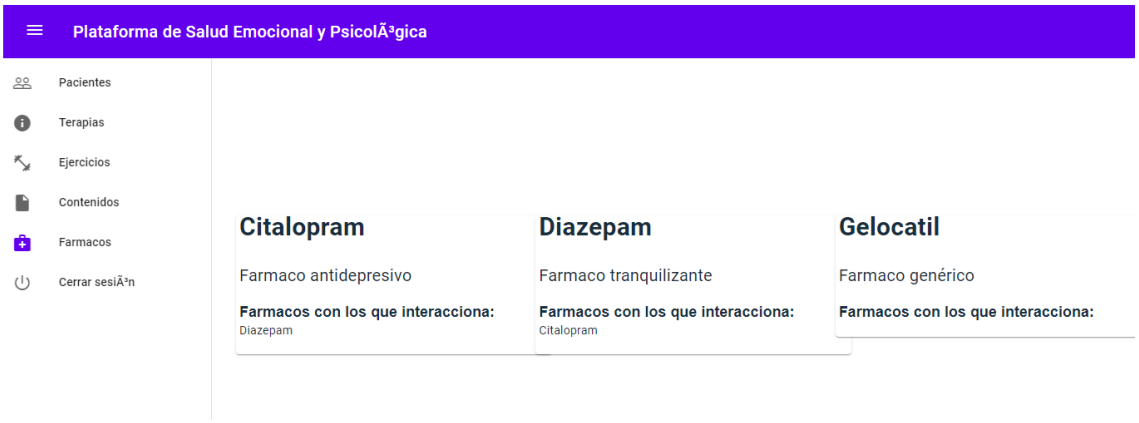

Ilustración 21Plantilla de visualización de los fármacos

La siguiente plantilla, Ilustración 22, que se ha creado es la de "contenidoTerapias". En ella se observa la lista de terapias almacenada en la base de datos y la información asociada a cada una de estas terapias. Todo esto en formato tarjeta:

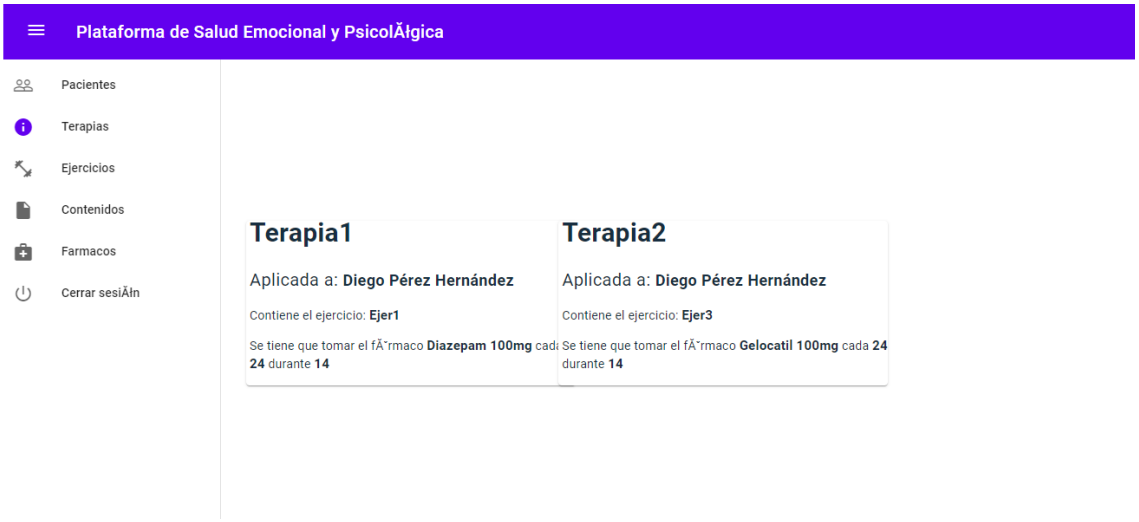

Ilustración 22Plantilla para ver las terapias

Para crear los ejercicios se tiene la plantilla de "contenidoCrearEjercicioPantalla" que está en la Ilustración 23:

Ilustración 23 Plantilla para crear ejercicios

Esta plantilla es similar a las creadas para crear los distintos tipos de contenido. Se tiene 3 entradas de texto por las que se escribe la información que se requiere para poder tener un ejercicio completo. Cuando se le pulsa al botón de "Crear Ejercicio" lleva al identificador de "crear_ejercicio".

Con la plantilla de "contenidoPantallaIncluirContenidos", lo que se representa es la lista de los contenidos de la base de datos en formato tarjeta, con toda la información de los contenidos y, dentro de cada tarjeta, un botón con el icono de "+" con el que, al pulsarlo, se añade el contenido al ejercicio que se le ha pasado como información a esta plantilla. Todo esto luce como en la Ilustración 24:

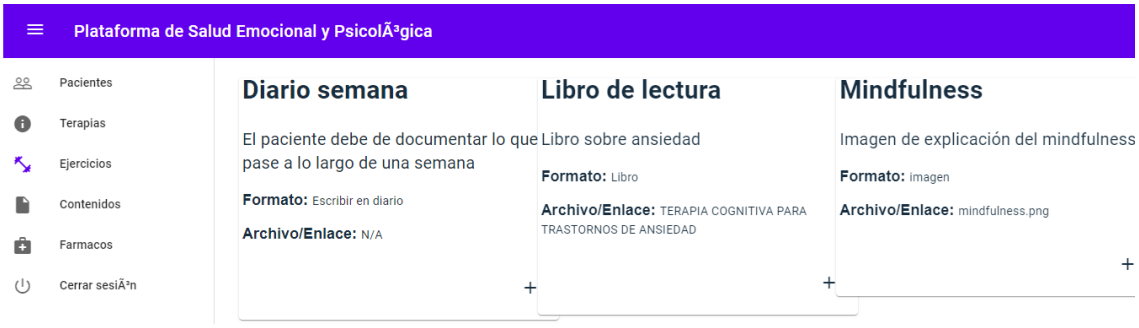

Ilustración 24 Plantilla con la que se incluyen los contenidos en un ejercicio

En "contenidoGraficos" se representa en dos gráficas, todos los datos de los sensores para un paciente ordenados de forma cronológica. También, debajo de la plantilla, se tiene dos calendarios que sirven para seleccionar fechas mínimas y máximas para acotar los datos. Con el icono con forma de gráfica que sirve como botón, lleva a "cargar_datos_acotados" con entrada del DNI como en la Ilustración 25.

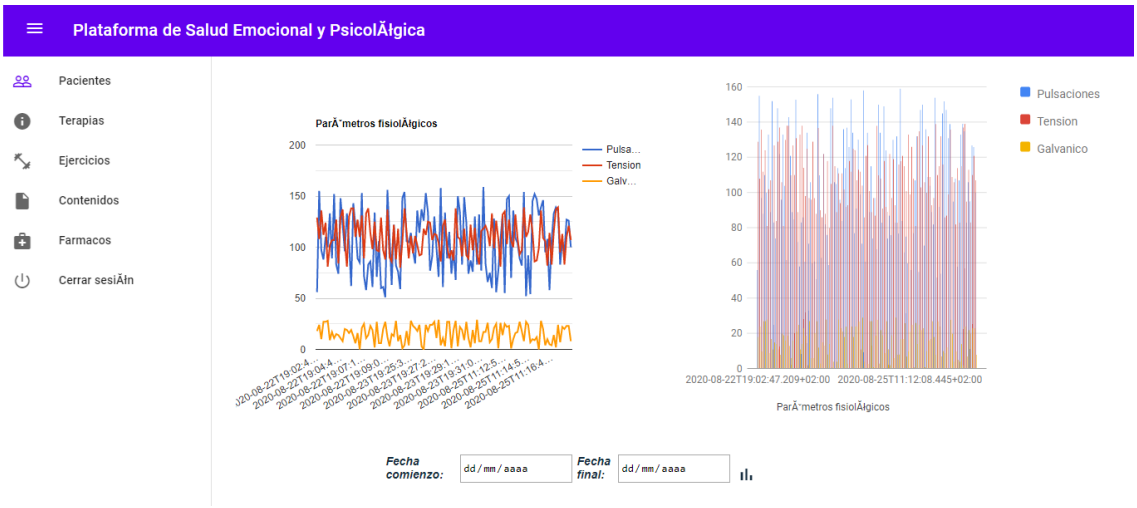

Ilustración 25 Plantilla de todos los datos de los sensores de un paciente

Con "contenidoGraficosAcotados" se tiene una plantilla con el mismo diseño que la plantilla explicada justo antes, pero, en vez de mostrar todos los datos se muestran los datos dentro de los márgenes que se han especificado en la plantilla anterior. En esta plantilla, también se pueden acotar de nuevo los datos si se desea.

Desde "contenidoPantallaHacerEjercicio", se ha diseñado una plantilla para hacer el ejercicio cuando se está midiendo mediante la aplicación de Google Cloud Vision API. El diseño de la plantilla es el de la Ilustración 26:

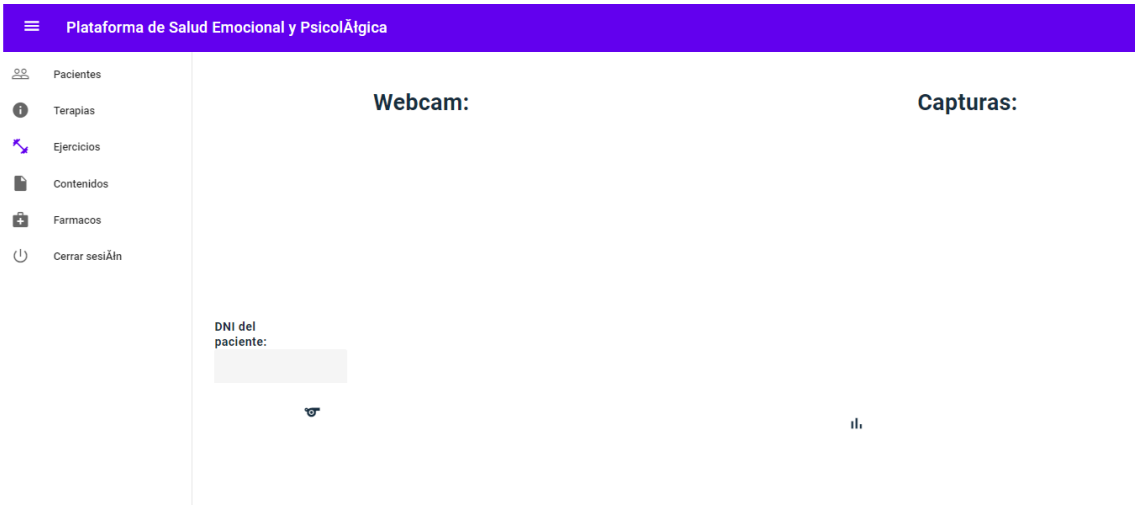

Ilustración 26 Plantilla con la que se hace el ejercicio con la medición de la aplicación de Google Cloud Vision API

En ella hay 3 elementos con los que se pueden interactuar. Estos son un cuadro en el que se puede introducir texto y dos botones, uno con un silbato como icono y otro con unas gráficas. El del silbato lleva a la función que ejecuta la llamada de la aplicación, entre otras cosas. El del icono de los gráficos se le debe pulsar cuando se vea conveniente, pero preferiblemente después de terminar el ejercicio, y este lleva al identificador de “resultados_ejercicios”.

Cuando se pulsa el botón que indica el comienzo del ejercicio (el icono del silbato) tras un breve tiempo se enciende la webcam y aparece el vídeo que esta grava al lado de “webcam”. Y, como para que la aplicación de Google funcione correctamente se le deben de suministrar imágenes, las capturas que se vayan haciendo durante el ejercicio se verán de forma continuada a la derecha de “Captura”.

En cuanto a “contenidoPantallaResultadosEjercicio” se ve como en la Ilustración 27:

Ilustración 27 Plantilla de los resultados de la aplicación de Google Cloud Vision API

En la que se muestra cada uno de los parámetros que se han calculado en el endpoint de "resultados_vision" de una forma más estructurada y con unas introducciones.

La plantilla de "contenidotensor" se ve como en la Ilustración 28:

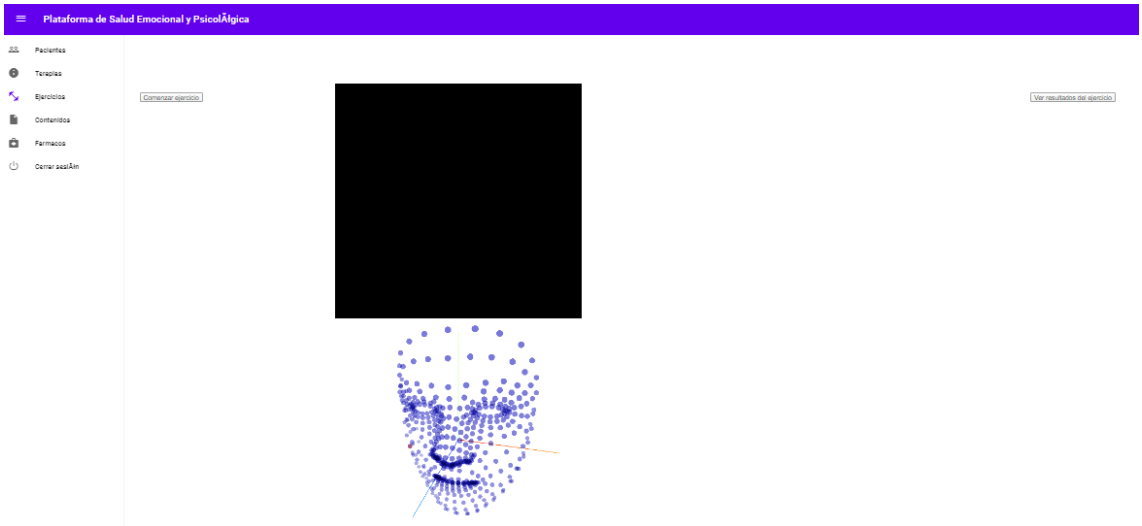

Ilustración 28 Plantilla para hacer un ejercicio con la medición del modelo de TensorFlow.

Esta plantilla, al principio, tanto donde está el cuadrado negro como donde está la mascarilla está en blanco. Pero, tras darle al botón de "Comenzar ejercicio" se llama a la función que inicia el programa que evalúa mediante el modelo de TensorFlow

la posición del rostro. Esto hace que al poco rato se enciende la webcam y apareza la máscara en 3D. El otro botón que aparece lleva al identificador de “vertensor”.

La última plantilla diseñada es “contenidovertensor”, la cual es similar a la Ilustración 29:

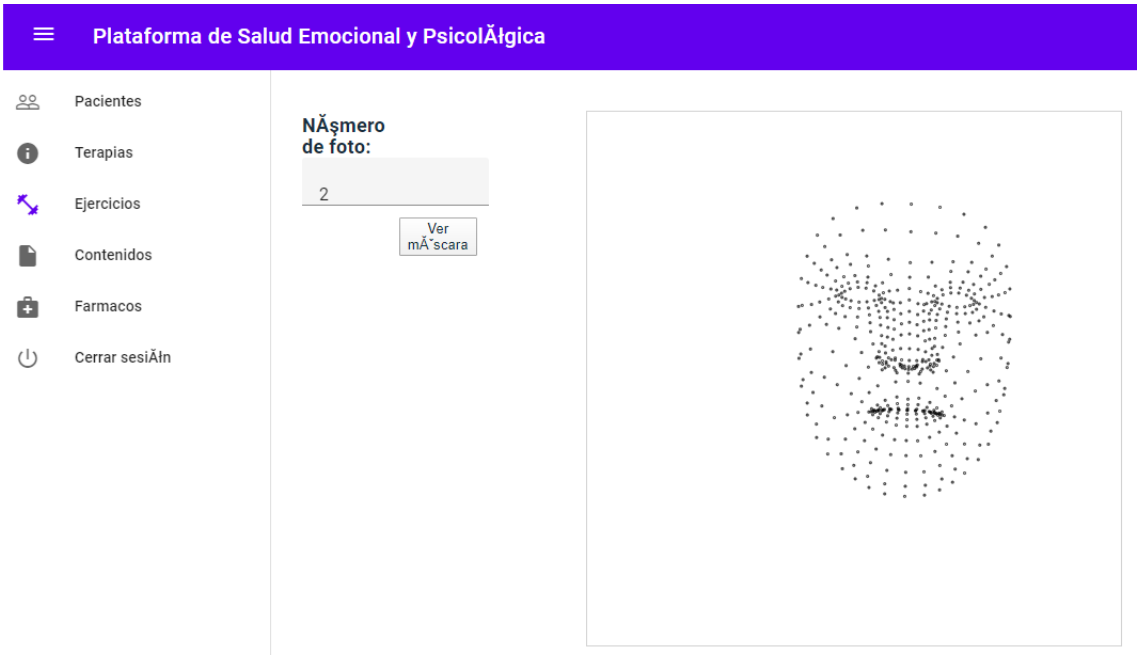

Ilustración 29 Plantilla para ver los resultados del modelo de TensorFlow

En la plantilla hay un cuadro de texto en el que se debe de escribir un valor numérico para ver la fotografía que se desee, un botón que lleva al identificador de “verresultadostensor” y un cuadro, donde se muestra la imagen de la máscara 2D de la fotografía deseada.

Una vez terminado la parte de las plantillas de la página, se ha programado las diferentes funciones. En estas funciones, principalmente lo que se hace es la llamada del endpoint y el tratamiento de los datos, si fuese necesario.

La primera de las funciones que se ha programado se trata de “cargar_pacientes”, desde la que se llama al endpoint “_lista_pacientes” y guarda en una variable la lista de los pacientes junto con la información y termina llamando a “templatePacientes” teniendo como entrada el contenido.

En la función “cargar_unpaciente” se llama al endpoint de “_cargar_paciente”. Luego, coge toda la información que se obtiene como respuesta y lo introduce como entrada a “templateUnPaciente”.

Para “cargar_ejercicios” trabaja de forma similar a la función de “cargar_pacientes”, pero, en este caso el endpoint con el que se trabaja es el de “lista_ejercicios” y cierra llamando a “templateCargarEjercicios”.

En “pantalla_elegir_tipo_ejercicio” el cual tiene como entrada el nombre del tipo de ejercicio, solamente llama a “templateElegirTipoEjercicio” con entrada del nombre del ejercicio.

La función “cargar_contenidos” llama a “templateContenidos”.

Con “cargar_contenidos_videos”, se usa el enpoint de “lista_contenidos_total”, y con un condicional, se comprueba que el tipo de contenidos sean de tipo vídeo. Si cumple, se almacena en una variable los datos y que se usan como entrada para llamar “templateContenidosVideos”.

Con “pantalla_crear_contenido_video” se llama a “templatePantallaCrearContenidoVideo”.

En “crear_contenido_video” se almacenan en distintas variables los valores que se han escrito en la página. Estos valores se usan como entrada en el endpoint de “incluir_contenido” y manda una alerta de la respuesta con el nombre del contenido o indicando que el contenido ya está incluido o es erróneo.

“cargar_contenidos_imagen” es igual que “cargar_contenidos_videos”, pero en este caso, la condición depende de que el tipo sea imagen, y antes de terminar la función se llama a “templateContenidosImagen”.

En “pantalla_crear_contenido_imagen” se llama a “templatePantallaCrearContenidoImagen”.

Para “crear_contenido_imagen” se usan los valores introducidos en la página, como en “crear_contenido_video”, se llama al mismo endpoint y al igual que el otro, se manda una alerta si fuese necesario.

En “cargar_contenidos_otros”, “pantalla_crear_contenido_otros” y “crear_contenidos_otros” se trabaja de forma similar al de los apartados anteriores pero en este caso no se restringe a tener que ser solo del tipo vídeo o imagen, sino que puede ser cualquiera.

Con la función de “cargar_farmacos” se llama al enpoint de “lista_farmacos_total” y se usa la respuesta para introducirlo en una variable que se introducirá como entrada en “templateFarmacos”.

En “cargar_terapias” se usa el enpoint “lista_terapias” y la respuesta de dicho enpoint se introduce en una variable, la cual se usa como entrada al llamar a “templateTerapias”.

Las tres siguientes funciones que se han diseñado: “cargar_datos”, “ejercicios” y “crear_ejercicio_pantalla” llaman cada una de ellas a su respectivas variables template: “templateDatos”, “templateEjercicios” y “templateCrearEjercicoPantalla”.

La función “crear_ejercicio” se asemeja a las otras funciones que también se basaban en crear, es decir, lo que hace es coger la entradas escritas en la página

web y usa esas entradas para llamar al enpoint de “crear_ejercicio” y al terminar el endpoint, salta un mensaje de alerta con el nombre del ejercicio.

En “pantalla_incluir_ejercicio” a la que se le suministra como entrada una variable del nombre del ejercicio, se llama al enpoint de “lista_contenidos_total” y se usa tanto la respuesta del enpoint como la variable de entrada de esta función para usarlo como entrada y llamar a la variable “templatePantallaIncluirContenidos”.

La función “incluir_contendio_en_ejercicio” que tiene dos variables de entrada, nombre del ejercicio y nombre del contenido, se usa el enpoint “incluir_contenido_en_ejercicio” y se manda un mensaje de alerta con la respuesta del enpoint.

“eliminar_ejercicio” con el endpoint “borrar_ejercicio” al cual se le mete como entrada, la variable que viene con esta función, se le llama ha dicho endpoint y salta un mensaje de alerta de la respuesta del endpoint.

En la función “cargar_graficos” con entrada del DNI del paciente del cual se quieren visualizar los datos de los sensores del paciente. Primero se cargan los paquetes de Google Chart. Luego, hay dos funciones, una para cada tipo de gráfica que se quiere diseñar, y se programa todo lo necesario para que estas funciones, entre lo que se incluye una llamada al endpoint que permite visualizar los datos de los sensores “lista_datos”, y también se llama a una variable que representa la visualización en el documento. Por último, al terminar las dos funciones del diseño de los gráficos, se llama al enpoint de “cargar_un_paciente” y con su respuesta se almacena en una variable con la que se llama a la variable de “templateGraficos”.

“cargar_datos_acotados”funciona igual que la función anterior, pero en vez de llamar al enpoint de “lista_datos”, se llama a “lista_datos_acotados”. Los parámetros de fechas máximos y mínimos se obtienen directamente del documento en el que se han introducido las fechas.

Desde la función “pantalla_hacer_ejercicio” con variable de entrada el nombre del ejercicio que se quiere hacer, se llama al enpoint “ver_ejercio” con la variable de entrada de esta función. Al terminar el enpoint, se llama a la variable “templatePantallaHacerEjercicio” y se le pasa como entrada la respuesta del endpoint.

La siguiente función, a la que se le ha llamado “capture” es la que hace la aplicación de Google Cloud Vision API. En esta función, primero se enciende la webcam y se empieza a mostrar por pantalla en la posición que la página de diseño indica. Luego, con un condicional se empieza a contar tantos ciclos haga. El condicional depende de la duración del ejercicio, el cual se pasa como entrada. A continuación, se hace una captura de lo que la webcam esté grabando y lo pone en el formato base64, el cuál necesita la aplicación para funcionar. Con esta imagen se llama a la aplicación de Google habiendo añadido también anteriormente la autorización pertinente para que la aplicación funcione. Las respuestas que de la aplicación se guardan en distintas variables las cuales se usan como entrada al llamar al endpoint

“insertar_datos_vision”. Por último, se espera un tiempo y vuelve a empezar la función. Cuando todo el ciclo se complete se muestra como que el ejercicio se ha completado.

Con la función “pantallacloudvision” se llama a la variable tipo template de “templatePantallacloudvision”.

Desde “resultados_ejercicio”, primero, con el DNI que se obtiene del propio documento se hace una gráfica de forma similar a la que se ha hecho en las funciones de “cargar_graficos” y “cargar_datos_acotados”. Después del diseño de los gráficos, se llama al endpoint de “resultados_vision” y se guarda la respuesta en una variable que se usa como entrada para “templatePantallaResultadosEjercicio”. Pero antes de llamar a esta última variable de tipo template, se llama al endpoint “borrar_datos_vision”.

Al llamar a “logout” se vuelve al .html de inicio.

Luego, siguen las funciones que se utilizan para trabajar con el modelo de TensorFlow, que están explicadas en el punto 3.2.6. y las funciones de “tensor” y “vertensor” que llevan a la variable tipo template de “templatetensor” y “vertensor”respectivamente.

La última función es “verresultadostensor” que obtiene valores de entrada de la página web para usarlos como entrada en el endpoint “ver_datos_tensorflow_individual” con el que se tiene la máscara del modelo y la dibuja en la página web antes de terminar la función.

Tras terminar con todas las funciones, se han diseñado las variables de tipo template para poder enlazar con las plantillas del diseño de la pagina web.

Y, por último, se han creado los identificadores o lo que manejan las distintas opciones de la página web mediante los pulsadores. La mayoría de las veces se accede directamente a la función, pero en alguna ocasión se usan los identificadores.

- **paciente:** Es similar al .html del terapeuta, pero reduciendo las posibilidades que tienen respecto a este. La plantilla principal es la que se observa en la Ilustración 30:

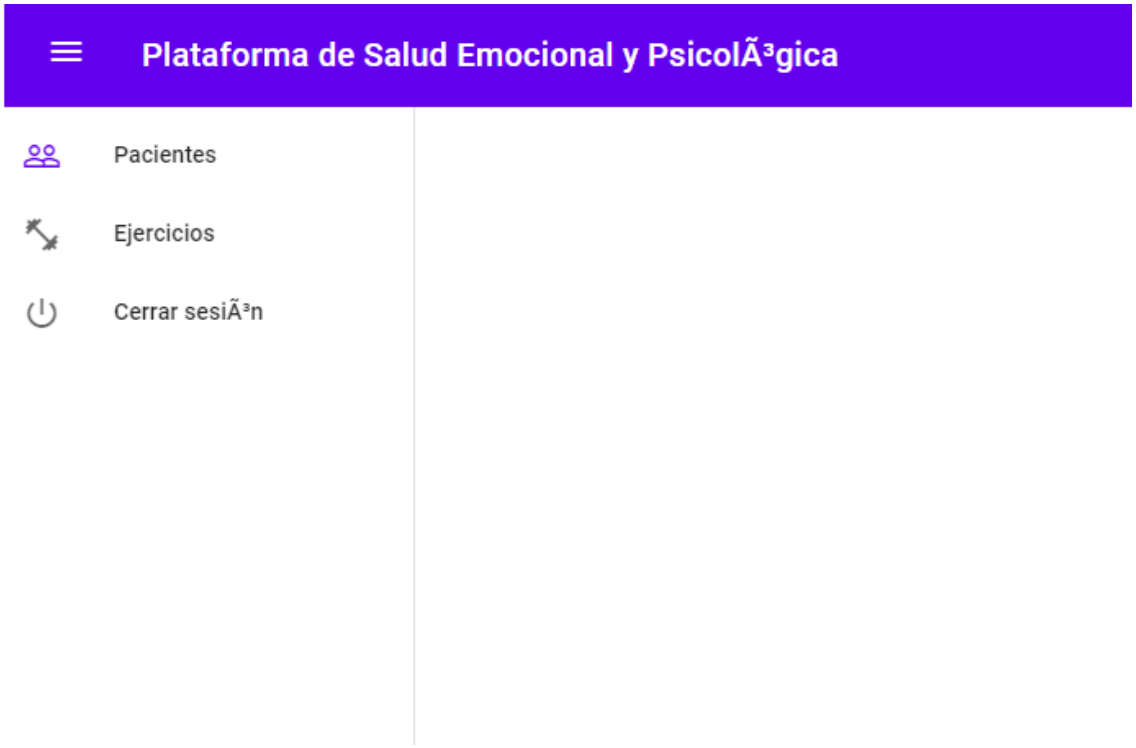

Ilustración 30 Plantilla de inicio de paciente

Como se puede ver, solo tiene la opción de “Pacientes, “Ejercicios” y “Cerrar sesión”

En la primera de estas solamente puede acceder a su información y ver los datos de los sensores de sus propiedades fisiológicas. Las otras dos opciones son iguales a las que están diseñadas en terapeuta, pero en ejercicios solo se pueden ver los ejercicios y hacerlos, pero no crearlos.

La diferencia principal frente al terapeuta se basa en que en la parte del desarrollo de las diferentes funciones, al principio, se llama al endpoint de “ver_ultimo_registro”. Con este endpoint, se obtiene la información del sujeto que se acaba de logear y se crea una variable DNI en la que se guarda la información del DNI que da como salida el endpoint.

Esta variable se mantiene fija a lo largo de todo el tiempo que se mantenga uno en la pagina web, por lo que cada vez que se acceda a una función en la que sea necesario diferenciar por el DNI, se usará siempre el DNI de quien se ha logeado, y de esta manera se consigue que solo pueda ver su propia información.

5. ANÁLISIS DE RESULTADOS

Para comprobar el correcto funcionamiento de la aplicación que se ha desarrollado se va a proceder a la creación de un ejercicio y la posterior realización de dicho ejercicio todo desde el punto de vista de un terapeuta. Todo se mostrará en imágenes paso a paso, desde el inicio de sesión y teniendo en cuenta que ya está el servicio trabajando.

1. Se hace el inicio de sesión, Ilustración 31, siendo el correo y la contraseña, perteneciente a un terapeuta insertado previamente:

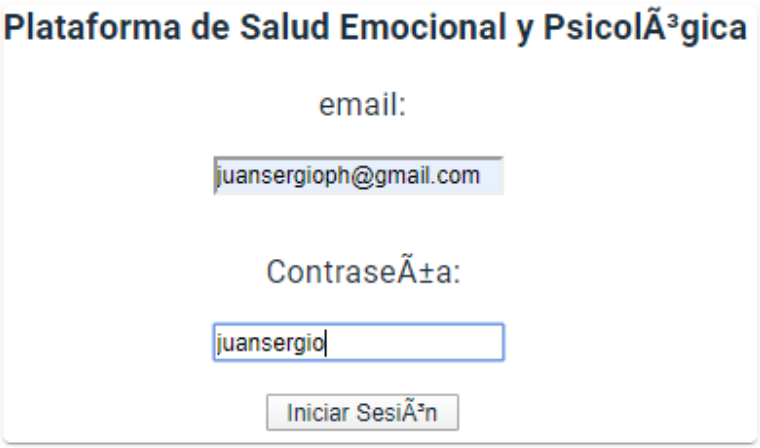

Ilustración 31Inicio de sesión como terapeuta

2. Tras pulsar el botón de Iniciar Sesión aparece la pagina principal del terapeuta como se ve en la Ilustración 32:

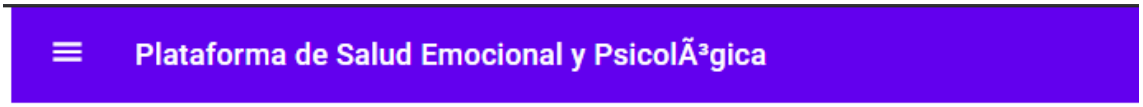

Ilustración 32 Página de principal del terapeuta

3. Luego, se le pulsa en el desplegable (las tres líneas de arriba a la izquierda) y se despliegan una serie de opciones:

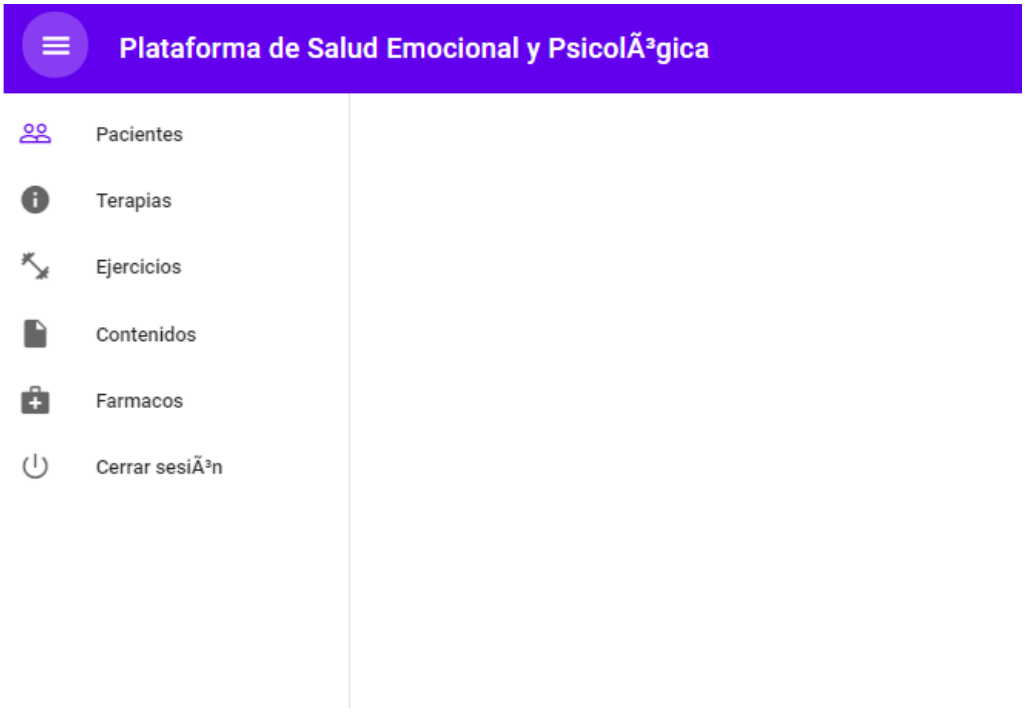

Ilustración 33 Página principal con las opciones

4. Como se quiere crear un ejercicio, primero veremos todos los ejercicios que hay ya creados. Para ello, tras pulsar en Ejercicios nos aparecerán dos opciones, como se ve en la Ilustración 34 y se elige la primera de ellas:

Ilustración 34 Elegir entre ver o crear ejercicios

5. Y se observan los ejercicios que aparecen en la Ilustración 35:

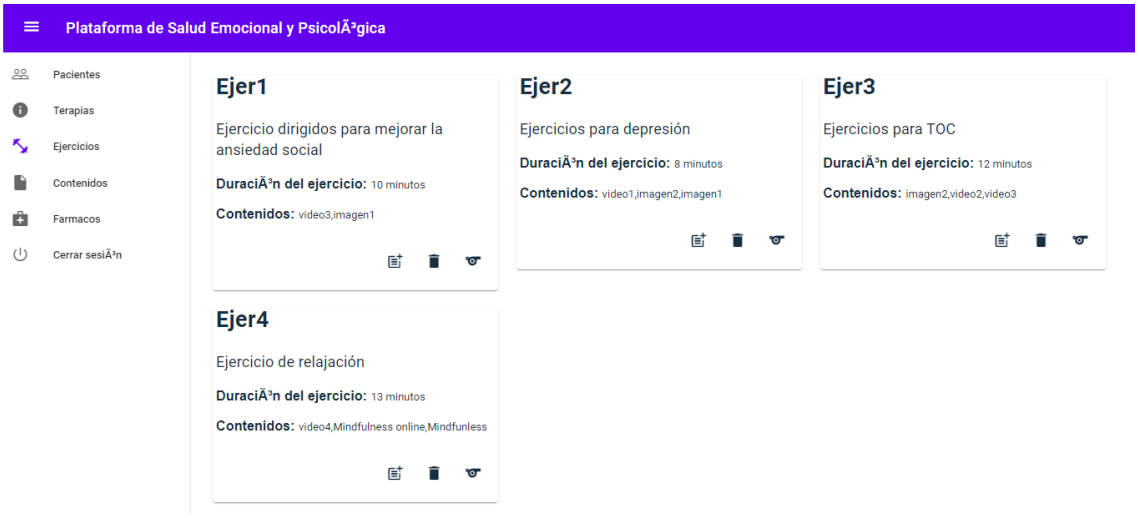

Ilustración 35Lista de ejercicios

6. Tras ver la lista de ejercicios, se le pulsa de nuevo al botón de Ejercicios para volver a lo observado en la Ilustración 34 y esta vez se elige la segunda opción, la de CREAR EJERCICIO NUEVO y aparecerá la imagen que se ve en la Ilustración 36. También, se completarán los campos necesarios para crear el ejercicio como se observa en la misma Ilustración:

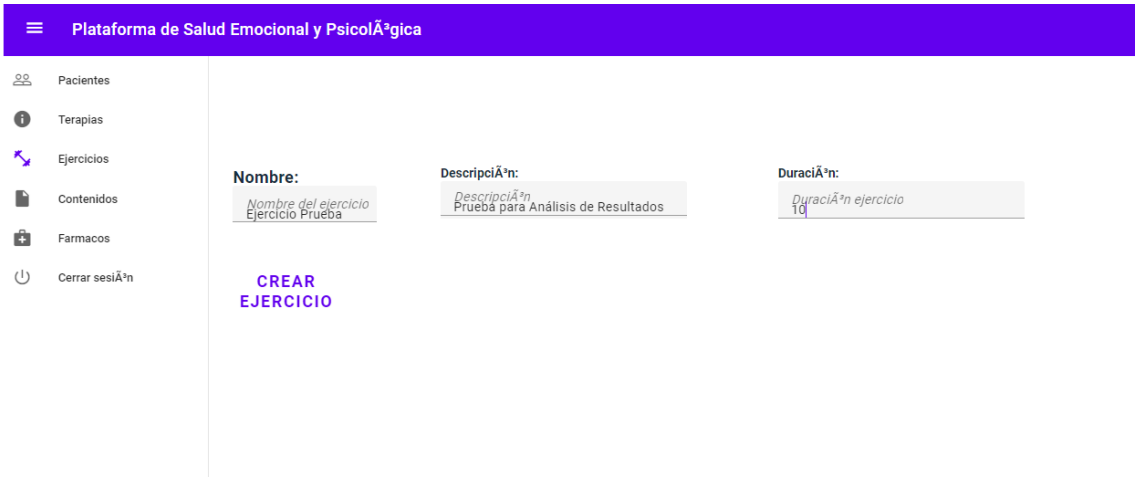

Ilustración 36 Creación de un ejercicio

7. Tras pulsar en el botón de CREAR EJERCICIO se mostrará un mensaje indicando que se ha creado correctamente como se ve en la Ilustración 37:

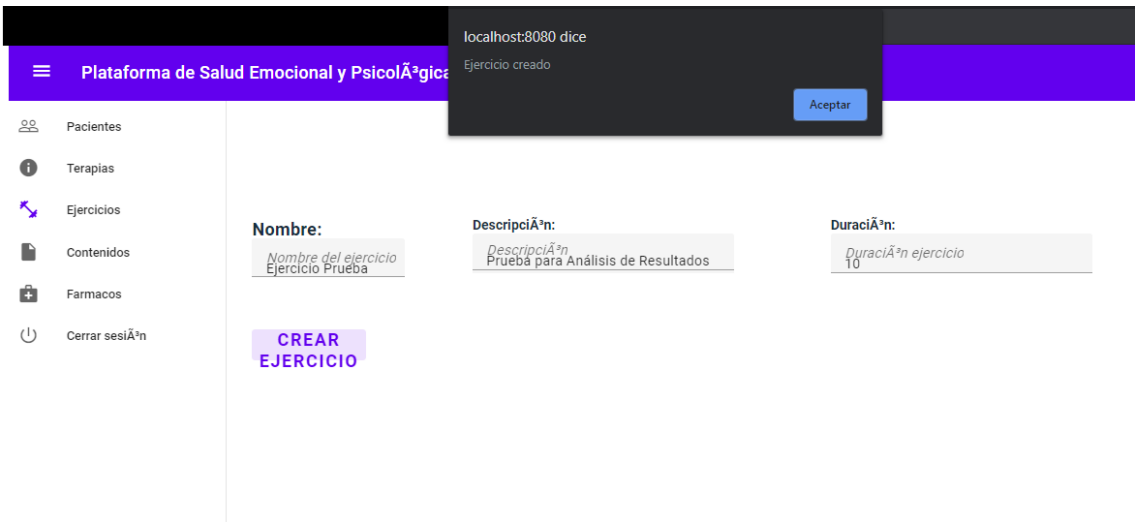

Ilustración 37 Creación de ejercicio con éxito

8. Tras crear correctamente el ejercicio, se le vuelve a pulsar al botón de Ejercicios y de nuevo a VER LISTA DE EJERCICIOS, y ahora se observa en la Ilustración 38 como aparece el ejercicio recién creado:

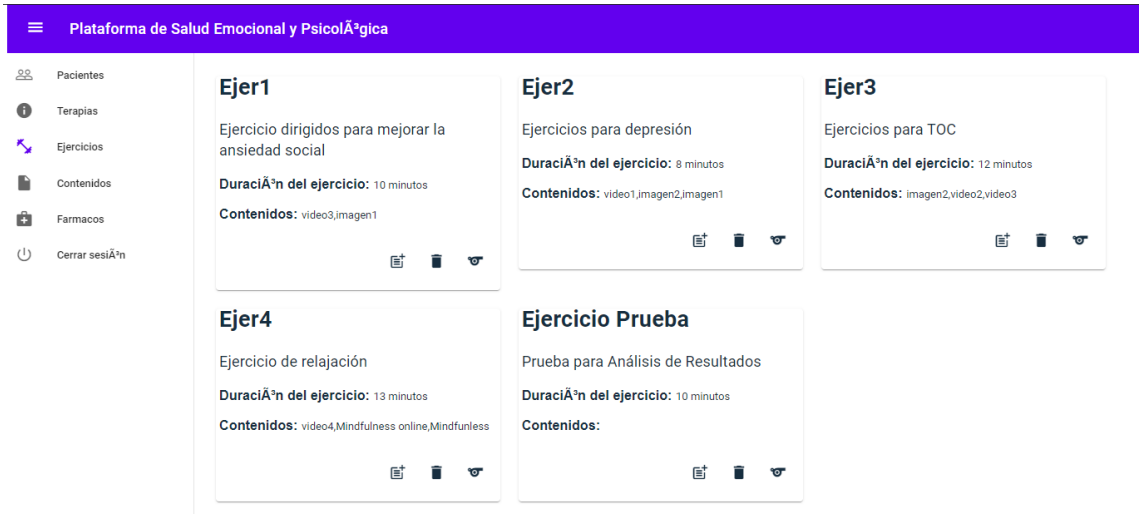

Ilustración 38 Lista de Ejercicios con nuevo ejercicio creado

9. Tras crear el ejercicio, se puede ver que no tiene contenidos, por lo que se le pulsa al primer botón dentro de la tarjeta del ejercicio creado y, como muestra la Ilustración 39, se observan todos los contenidos previamente creados:

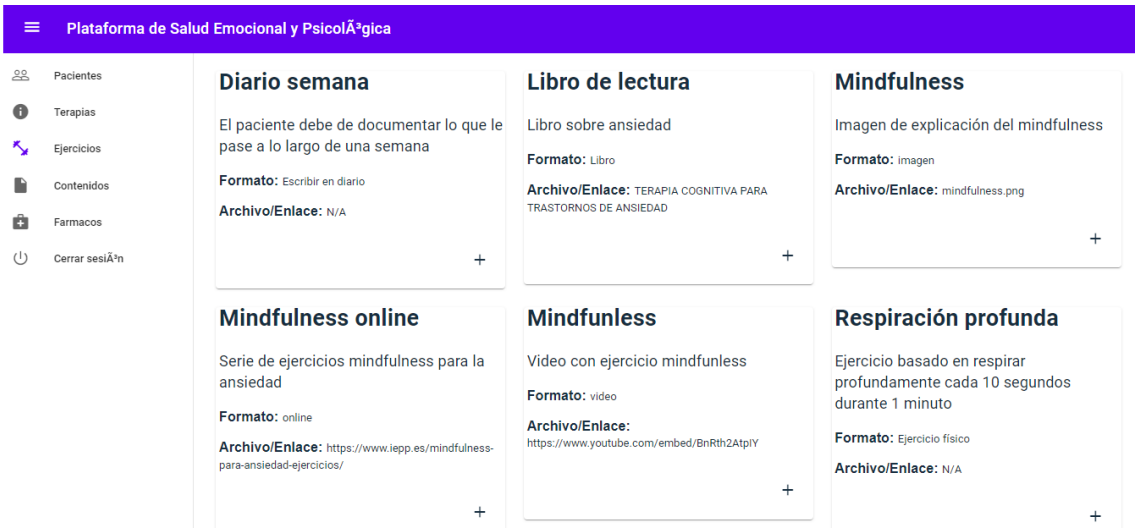

Ilustración 39 Lista de contenidos para incluir en el ejercicio

10. Para esta prueba, se le pulsará al botón + de los dos primeros contenidos para incluirlos en este ejercicio y cada vez que se le pulsa a cada uno de ellos aparecerá un mensaje diciendo que se ha añadido correctamente:

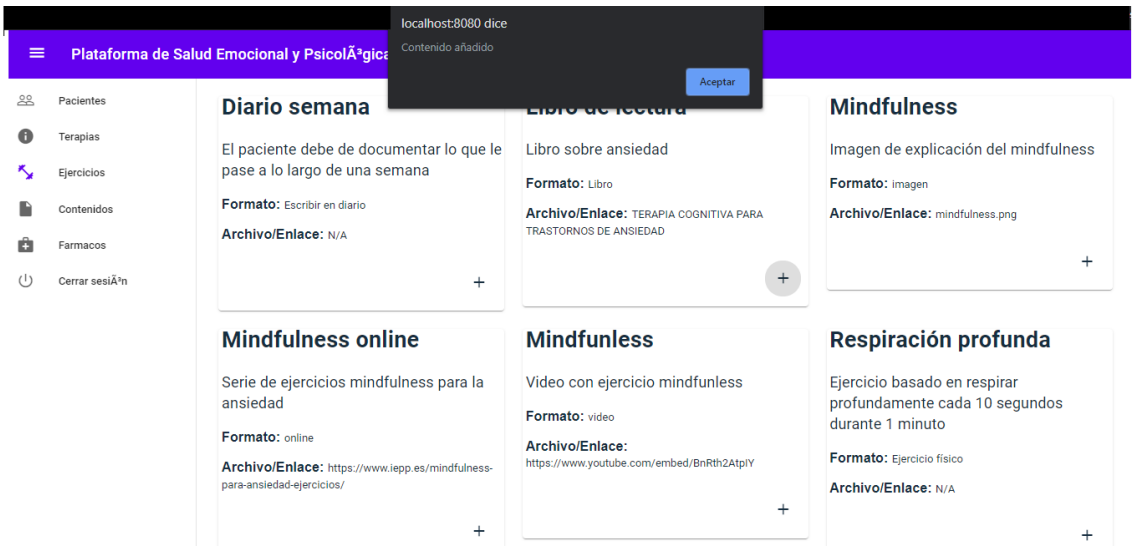

Ilustración 40 Añadido contenido con éxito

11. Tras haber añadido ambos contenidos con éxito, se vuelve otra vez a la lista de los ejercicios como se ha hecho anteriormente y se observa en la Ilustración 41, en el ejercicio creado como aparecen estos dos nuevos contenidos:

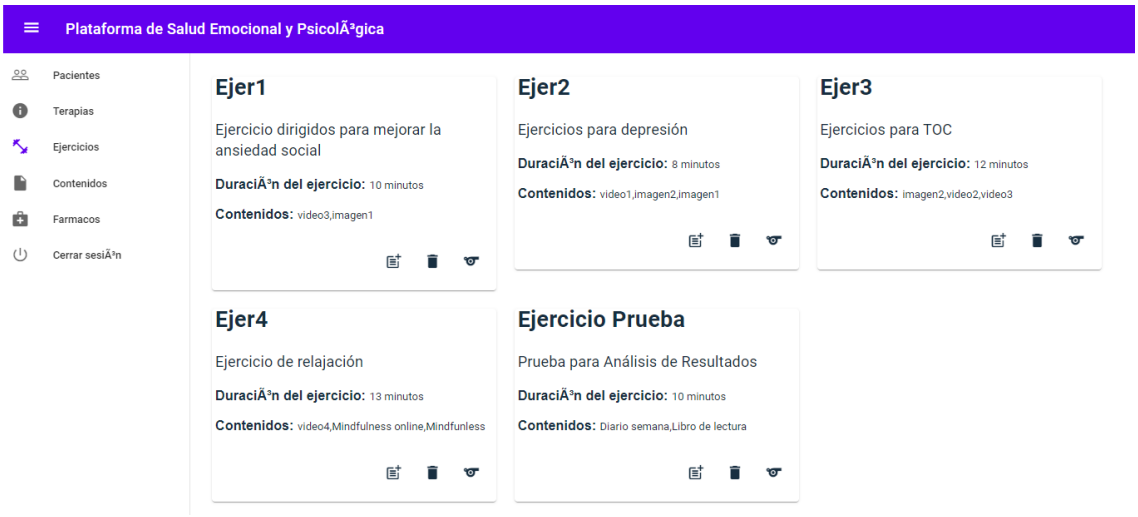

Ilustración 41 Lista de ejercicios, con el ejercicio de prueba y los contenidos añadidos

12. A continuación, para realizar el ejercicio se le pulsa al botón del silbato del ejercicio creado que lleva a una página para seleccionar el tipo de ejercicio como se ve en la Ilustración 42:

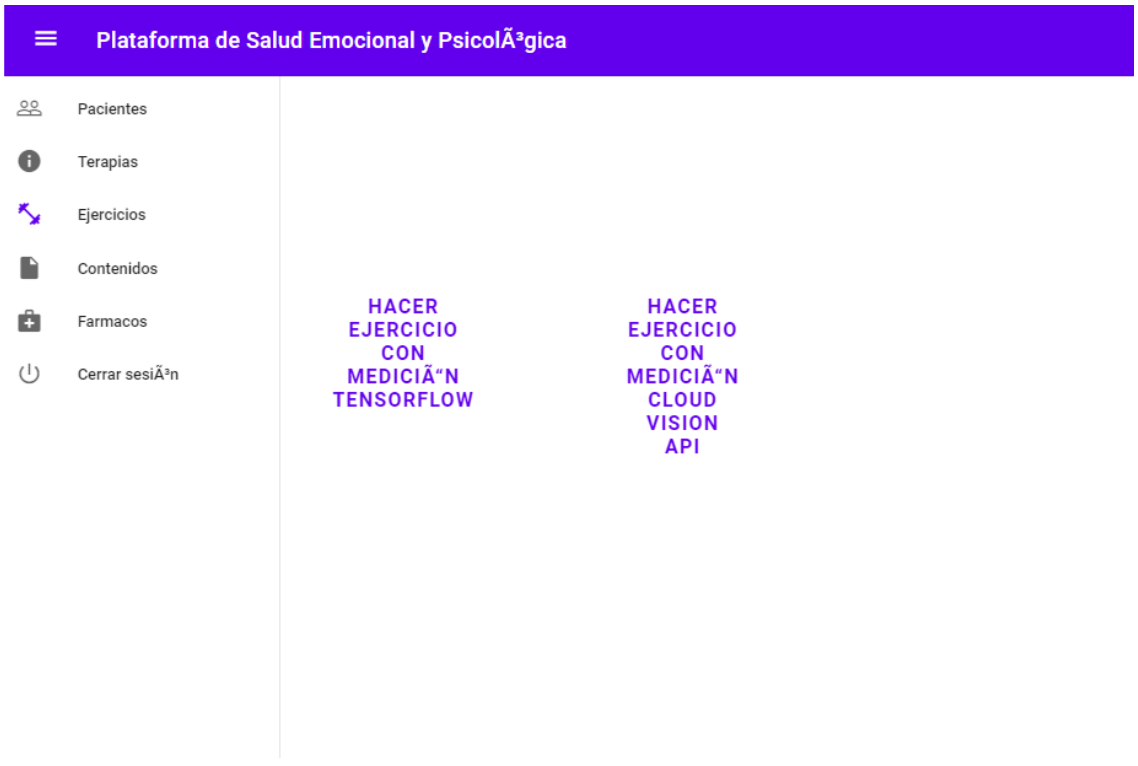

Ilustración 42 Seleción de medición del ejercicio

13. Para esta prueba, se selecciona el botón de HACER EJERCICIO CON MEDICIÓN TENSORFLOW, y este nos lleva a la página de espera para hacer el ejercicio:

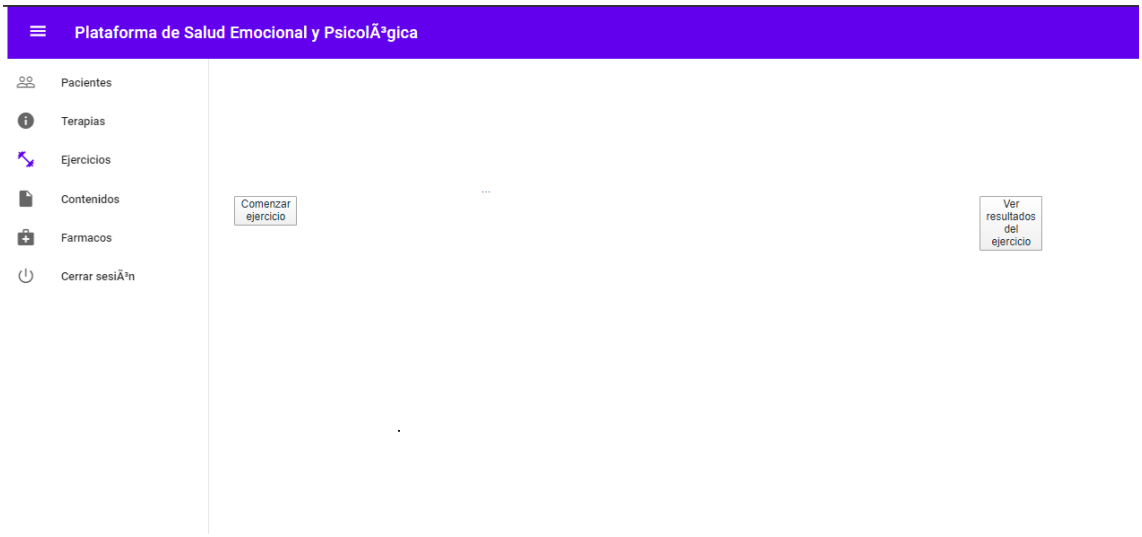

Ilustración 43 Página de espera para hacer el ejercicio mediante medición con el modelo de TensorFlow

14. Cuando el paciente esté preparado para hacer el ejercicio, le debe de pulsar al botón de Comenzar ejercicio y empezará a medir las marcas del rostro del paciente como se observa en la Ilustración 44:

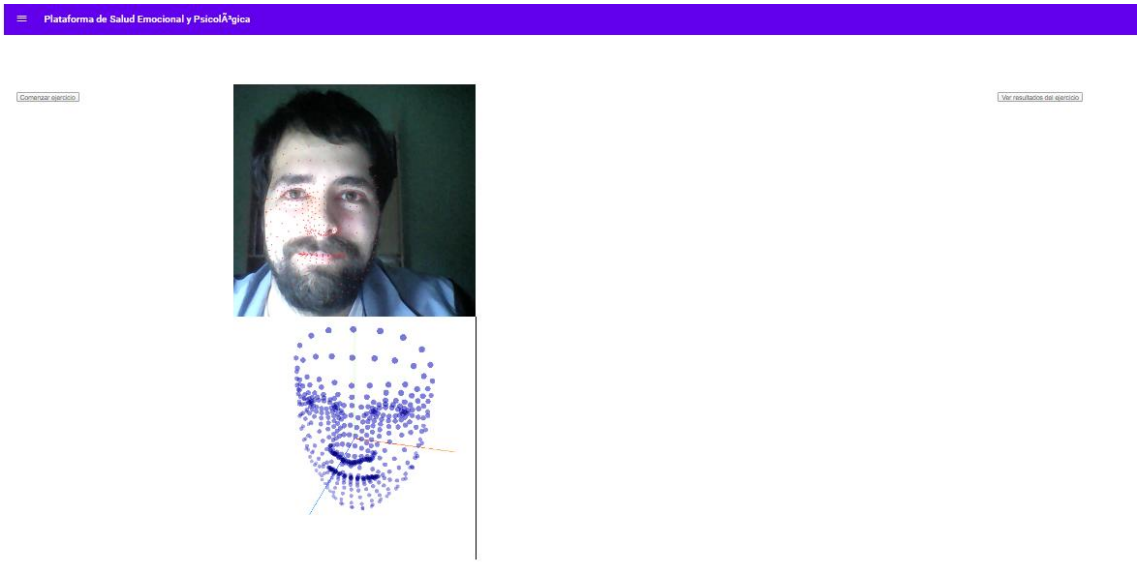

Ilustración 44 Paciente haciendo el ejercicio

15. Tras terminar el ejercicio, se le pulsa al botón de Ver resultados del ejercicio que llevará a la pagina de la Ilustración 45:

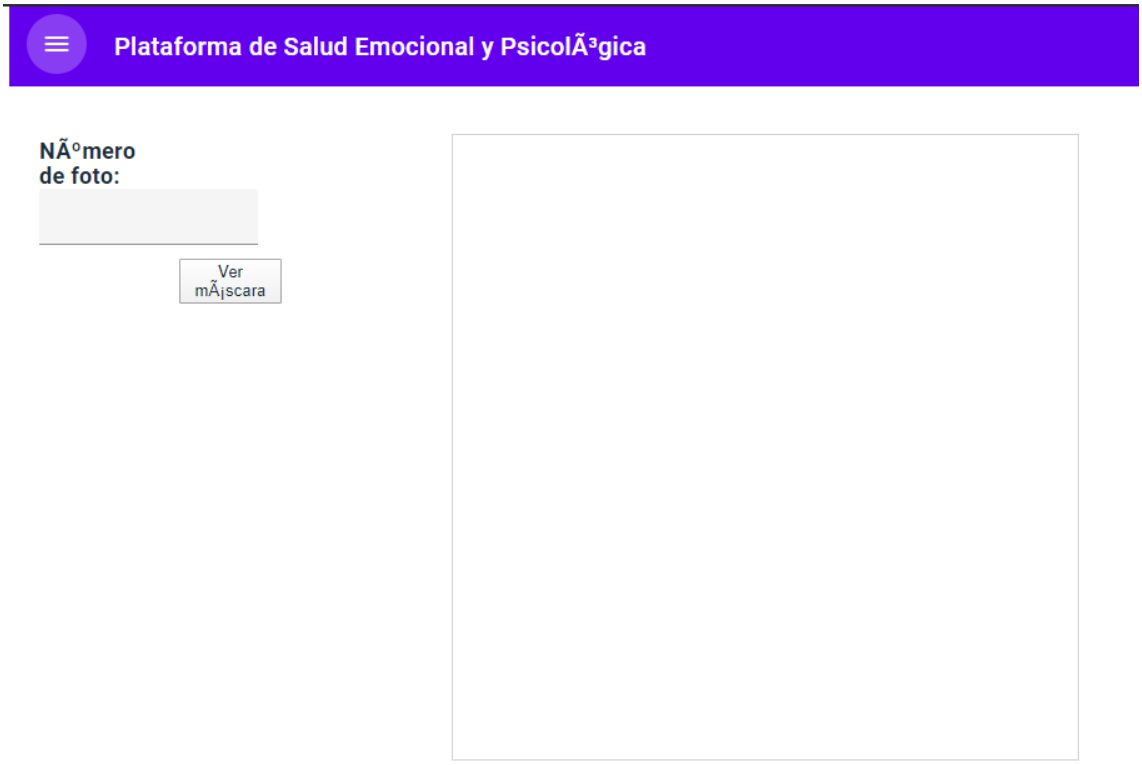

Ilustración 45 Página de espera para ver los resultados de los ejercicios

16. Para ver cada una de las máscaras que se han ido haciendo a lo largo del desarrollo del ejercicio tan solo hay que escribir el número de la máscara y darle al botón de Ver máscara y aparecerá la máscara del paciente. En la Ilustración 46 se muestran la máscara 1 y 2 del paciente apareciendo en la primera tranquilo y con la boca cerrada y en la segunda asustado y con la boca abierta:

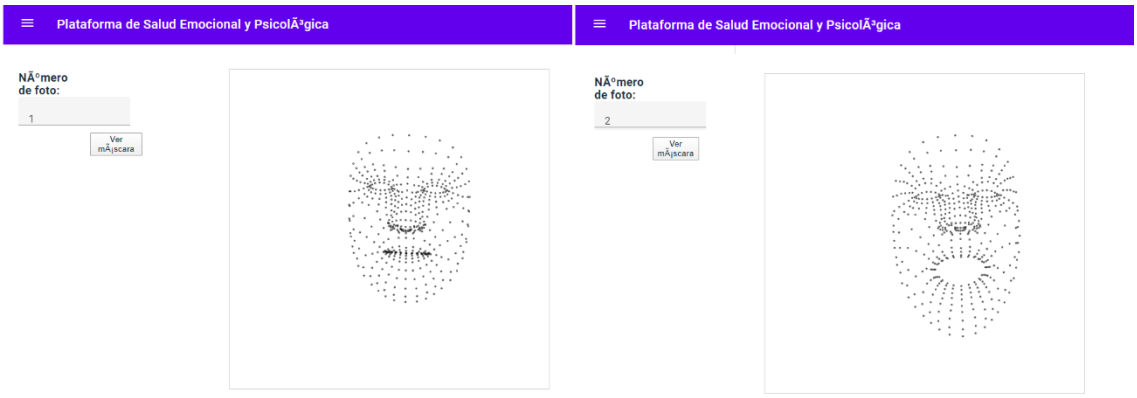

Ilustración 46 Resultados del ejercicio

17. Tras esto, el terapeuta, si quiere observar el historial de los gráficos de los sensores del paciente, le pulsa al botón del paciente y aparecen la lista de los pacientes como se ve en la Ilustración 47:

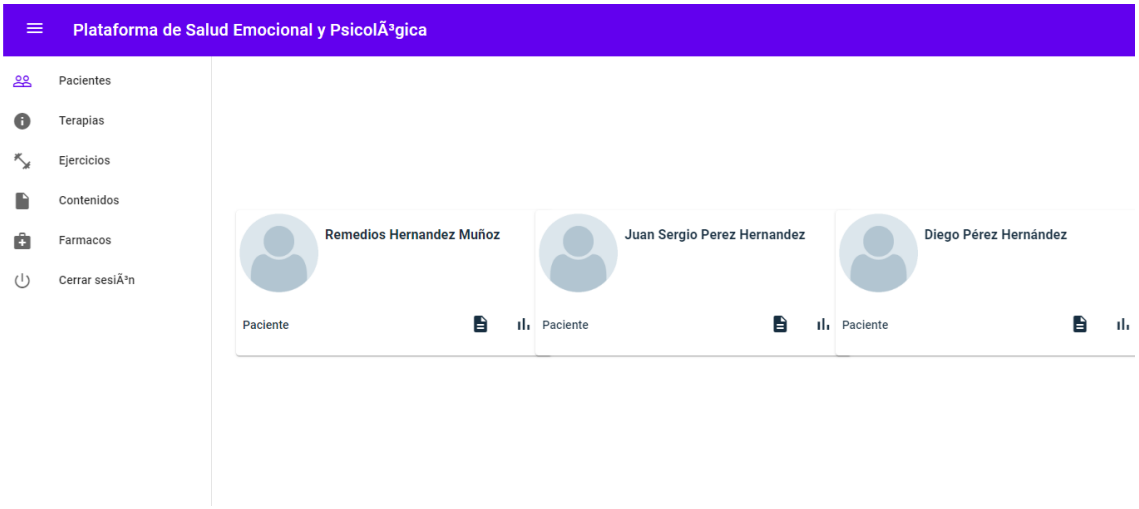

Ilustración 47 Lista de pacientes

18. Luego, le pulsa al icono de las gráficas del paciente del que quiera observar los datos de los sensores y se observan las gráficas que se ven en la Ilustración 48 para esta prueba:

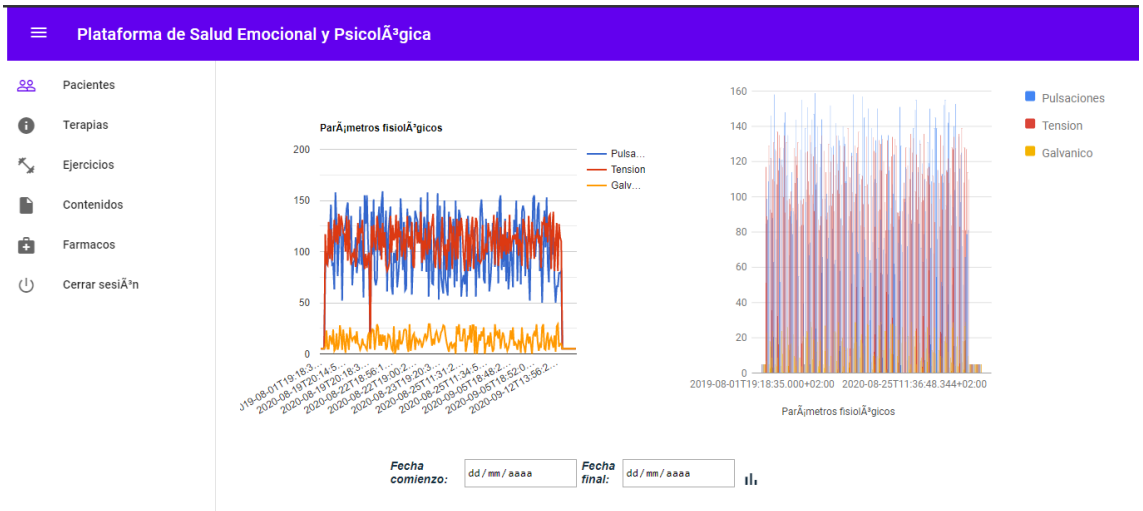

Ilustración 48 Gráficas de los datos de los sensores

19. Para acotar los datos, se ponen las fechas límites con el calendario desplegable que y se le pulsa al icono de las gráficas y estas se acotan como se ve en la Ilustración 49 en la parte izquierda, y también se representa el histórico para poder comparar en la parte derecha:

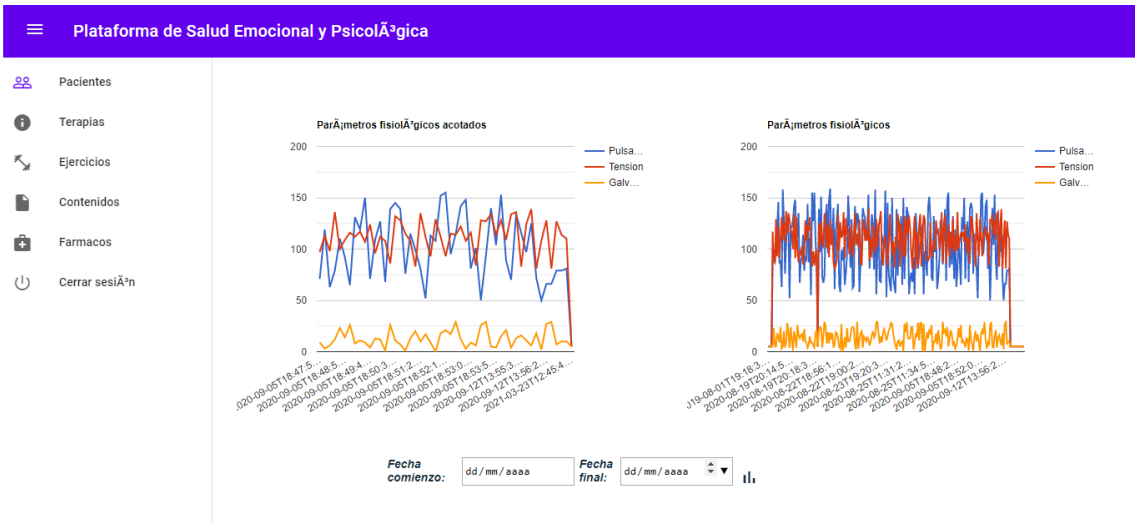

Ilustración 49 Gráficas acotadas

20. Cuando el terapeuta termina, le pulsa al botón de Cerrar sesión y sale de esta página para volver a la página de inicio de sesión como se ve en la Ilustración 50:

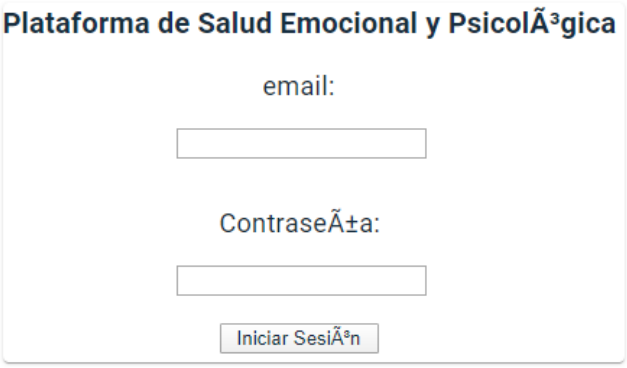

Ilustración 50 Vuelta a la página inicial tras cerrar sesión

Esto sería una prueba del correcto funcionamiento de la aplicación web, pero también hay que destacar otros aspectos relevantes en cuanto al resto de aplicaciones y desarrollo de todo el proyecto.

De cara al uso de los endpoints, es un formato muy fácil de aprender y útil en cuanto a trabajar de forma online, y gracias a la librería adaptada de Objectify se puede conseguir un almacén de datos de una forma muy cómoda.

El programa creado para el suministro de los datos de los sensores hace la función de representar unos datos y sirven para implementar la herramienta de adquisición de datos que se ha desarrollado, pero en un futuro, este programa se debería sustituir por los sensores que dan esta información sin necesidad de recurrir a modelizar los teóricos datos que deberían aparecer.

En cuanto a los modelos de IA usados, ambos estaban previamente entrenados lo que facilita su uso, pero los restringe mucho, es decir, son muy poco flexibles. Por ejemplo, el de Google da información muy valiosa, como es el estado anímico del paciente, pero suele ser muy poco fiable. Además, el tener que usar el formato de imagen en base64, al menos en la base de datos usada, imposibilita almacenar la foto porque ocupa demasiado espacio. También, es una aplicación muy lenta, del orden de 2 segundos hasta incluso 10 segundos dependiendo sobre todo de la velocidad a internet y lo saturada o no que esté la aplicación de Google en el momento de llamar a la aplicación. Otro problema que tiene esta aplicación es que no es un recurso gratuito. La primera vez que lo pruebas se proporcionan 3 meses gratuitos, siempre que no se pase un límite de llamadas a la aplicación, pero tras estos 3 meses, por cada llamada al recurso se debe de pagar.

Por otra parte, el modelo de TensorFlow, aún siendo un modelo ya previamente entrenado, para el caso de lo que se quiere desarrollar se asemeja más a una hoja en blanco, ya que, como información, solo proporciona la máscara. Esto hace que sea muy cómodo de ampliar y usar para trabajos futuros. También, tiene una ventaja y/o inconveniente, dependiendo de como se mire, que es que se trata de un modelo predictivo, es decir, que necesita una serie de imágenes anteriores y el modelo predice donde estará en el siguiente instante. Esto hace que sea necesario tener un vídeo, o en este caso, una grabación en directo, y al querer ir mostrando capturas de momentos concretos, este trabajo en segundo plano ralentiza mucho la página. Estas capturas se quieren hacer dado la imposibilidad de almacenar todos los datos que da como resultados a la velocidad que el modelo los da. También decir, que se trata de un modelo muy fiable y junto con su velocidad, hace que sea el modelo más atractivo de los dos usados.

Son muy importantes los desarrollos de aplicaciones online dado los tiempos que corren, y sobre todo los que apoyen a la salud mental. Que, aunque haya muchas aplicaciones de distinto tipo, pocas respaldan con datos fisiológicos del sujeto como se desarrolla en este proyecto.

Una vez que se han analizado los resultados del proyecto, se determinará si se han cumplido los objetivos propuestos con el siguiente apartado.

6. CONCLUSIONES

El primer objetivo que se había propuesto es el desarrollo de una herramienta de adquisición de datos, que funcione para visualizar y comparar datos adquiridos de sensores. Esta herramienta se ha conseguido desarrollar como se ha visto a lo largo del proyecto y se tiene una herramienta lo suficientemente robusta como lo es la aplicación web que sirve para visualizar todos los datos provenientes de los sensores de forma gráfica y los datos de ambos modelos de IA. Además, la herramienta también sirve para la creación, eliminación y modificación de distintas entidades que se han desarrollado a lo largo del proyecto para dar más opciones al terapeuta.

También se ha desarrollado una base de datos en la que se almacena toda la información necesaria. Esta base de datos se ha estructurado gracias al modelo de datos que se ha desarrollado y explicado, el cual ayuda a saber la relación entre las distintas entidades.

No se ha conseguido obtener los datos directamente de los sensores debido a la imposibilidad de obtener los sensores por culpa de la situación en la que nos encontramos ahora mismo (pandemia mundial debido al COVID-19), pero a cambio, se ha desarrollado una herramienta que simula los datos de los tipos de sensores que se iban a utilizar. Esta herramienta, a su vez, también tiene los elementos necesarios para almacenar los datos en la base de datos que se ha desarrollado, es decir, se tiene toda la infraestructura necesaria para poder hacer uso de la herramienta una vez se tengan los sensores.

La interfaz gráfica con la que interactúa el usuario también se ha podido desarrollar. Esta interfaz se ve a lo largo del proyecto viendo un ejemplo de como se ve en el punto 5. Análisis de Resultados.

En cuanto a los subobjetivos, el primero de ellos era la generación del modelo de datos el cual se explica en el punto 4.1. Modelo de Datos y en él se ve toda la estructura que siguen todas las entidades y como se relacionan entre ellas.

El segundo subobjetivo era el diseño de una página web. Esta página web es lo que ha servido como interfaz gráfica de cara al usuario para poder usar las distintas herramientas que se han desarrollado.

También se han podido usar dos modelos de IA. Uno de ellos se ha utilizado para medir puntos de la cara y comprobar el estado anímico del paciente y con el otro se ha podido generar una máscara con la que se pueden ver una gran cantidad de puntos de la cara con bastante fiabilidad.

Por último, en el punto 5. Análisis de Resultados se ha podido llegar a una conclusión de que modelo de los dos utilizados es mejor en que situaciones, se han visto una serie de ventajas e inconvenientes en cada uno de ellos, que en el siguiente apartado se comentarán para determinar como trabajar con ellos en el futuro.

7. TRABAJOS FUTUROS

El campo que desarrolla el trabajo es muy interesante y abre un gran abanico de posibilidades para trabajos futuros. Por ejemplo, en la página web en concreto, se puede implementar un sistema de inicio de sesión y gestión de usuarios como lo es Auth [13], que, con una base de datos, da permisos a usuarios de Google a acceder a los distintos servicios que se desarrollen en una aplicación como la del proyecto.

El diseño de la página es meramente orientativo, y sirve para representar todo lo desarrollado, por lo que, también se podría seguir aumentando el contenido de la página, dando más opciones o mejorando el diseño de las ya implementadas.

También, es interesante implementar físicamente los sensores y que no estén virtualizados como es el caso. Y ya que la herramienta de adquisición de datos ya ha sido implementada, sería bueno poder darle un uso real.

Con los modelos que se han usado: Google Cloud Vision API y el de TensorFlow. Con ambos se puede desarrollar un trabajo propio solo para cada uno de ellos sobre todo con el de TensorFlow. El de Google, es muy interesante y ya tiene muchas posibilidades debido a la gran variedad de opciones y los distintos tipos de resultados que tiene.

Con el modelo de TensorFlow obtenido, por como se ha implementado, se podrían llegar a usar las máscaras que se obtienen como resultados para entrenar otra IA que reconozca el estado anímico del sujeto, y así, no solo quedarse con lar libre interpretación que se dan a las máscaras. También, hay muchos otros modelos que son dignos de estudio dentro de la propia página de TensorFlow [10] y también se puede desarrollar una propia para que haga concretamente lo que uno necesita.

Siguiendo con los modelos de TensorFlow, no solo hay de máscaras, también hay otros que te permiten seguir las pupilas e incluso lo que el sujeto está observando, las cuales son muy interesantes para detectar cuan atento está el paciente, si le parece interesante o perturbador lo que está observando o haciendo.

Todo lo dicho es sobre todo de cara a la relación de paciente-terapeuta, pero también se puede llevar a muchos ámbitos como lo pueden ser médicos, nutricionales, deportivos o incluso en empresas, ya que la posibilidad de implementar el diseño de una página web a la que todos tengan acceso y a la que se le puedan suministrar diferentes entradas de sensores o predicciones de modelos tiene un gran abanico de posibilidades.

BIBLIOGRAFÍA

[1] Ministerio de Sanidad del Gobierno de España, «Bienestar Emocional,» 12 04 2020. [En línea]. Available: http://www.bemocion.mscbs.gob.es/.

[2] E. Vásquez-Dextre, «Mindfulness: Conceptos generales, psicoterapia y aplicaciones clínicas.,» *Revista de la Asociación Española de Neuropsiquiatría,* vol. 79, nº 1, 2016.

[3] R. Cowie, «Describing the emotional states that are expressed in speech,» *ELSEVIER,* vol. 10, nº 1-2, pp. 5-32, 2003.

[4] JSON, «Introducing JSON,» [En línea]. Available: https://www.json.org/json-en.html. [Último acceso: 31 Marzo 2021].

[5] «GitHub,» [En línea]. Available: https://github.com/objectify/objectify/wiki. [Último acceso: 20 Mayo 2020].

[6] Google, «Google Cloud Endpoints,» [En línea]. Available: https://cloud.google.com/endpoints/docs. [Último acceso: 1 04 2020].

[7] Google, «Material Design,» [En línea]. Available: https://material.io/. [Último acceso: 13 06 2020].

[8] Google, «Google Charts,» [En línea]. Available: https://developers.google.com/chart/interactive/docs. [Último acceso: 3 08 2020].

[9] The Code::Blocks team, [En línea]. Available: http://www.codeblocks.org/. [Último acceso: 10 08 2020].

[10] TensorFlow, «TensorFlow,» [En línea]. Available: https://www.tensorflow.org/learn?hl=es-419. [Último acceso: 26 12 2020].

[11] D. Stenberg, «curl,» [En línea]. Available: https://curl.haxx.se/. [Último acceso: 25 07 2020].

[12] JsViews, «jsviews,» [En línea]. Available: https://www.jsviews.com/. [Último acceso: 10 11 2020].

[13] Google, «Firebase Authentication,» [En línea]. Available: https://firebase.google.com/docs/auth?hl=es. [Último acceso: 2021 03 20].

[14] Google, «Google Cloud SDK,» [En línea]. Available: https://cloud.google.com/sdk/docs. [Último acceso: 1 04 2020].

[15] Google, «Google Cloud Vision API,» [En línea]. Available: https://cloud.google.com/vision/docs/. [Último acceso: 3 08 2020].

[16] J. Ellis, «github,» [En línea]. Available: https://github.com/hpssjellis/beginner-tensorflowjs-examples-in-javascript/tree/master/tfjs-models/facemesh/. [Último acceso: 10 11 2020].

[17] J. Gross, «The neural bases of emotion regulation: Reappraisal and suppression of negative emotion,» *ELSEVIER SCIENCE INC,* vol. 63, nº 6, pp. 577-586, 2008.

[18] P. Slovic, «Risk as analysis and risk as feelings: Some thoughts about affect, reason, risk, and rationality,» *Wiley,* vol. 24, nº 2, pp. 311-322, 2004.

[19] C. C. Tan y C. Eswaran, AUTOENCODER NEURAL NETWORKS, LAP Lambert Academic Publishing, 2010.

[20] C. Cao, Q. Hou y K. Zhou, «Displaced dynamic expression regression for real-time facial tracking and animation,» *ACM Transactions on Graphics,* nº 43, 2014.

[21] M. R. Milad, «Deficits in conditioned fear extinction in obsessive-compulsive disorder and neurobiological changes in the fear circuit,» *JAMA Psychiatry,* vol. 70, nº 6, pp. 608-618, 2013.

[22] C. A. Conde Cotes, L. C. Orozco Vargas, A. M. Báez Rangel y M. I. Dallos Arenales, «Aportes fisiológicos a la validez de criterio y constructo del diagnóstico de ansiedad según entrevista psiquiátrica y el State-Trait Anxiety Inventory (STAI) en una muestra de estudiantes universitarios colombianos,» *Revista Colombiana de Psiquiatría,* vol. 38, nº 2, pp. 262-278, 2009.