Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Grado en Ingeniería Informática

NeuroAutoTests: Aplicación web para el diagnóstico automático e inteligente del deterioro cognitivo

NeuroAutoTests: Web application for automatic and intelligent diagnosis of cognitive impairment

Francisco Jesús Mendes Gómez

La Laguna, 13 de septiembre de 2022

D. **Patricio García Báez**, con N.I.F. 43356987D profesor Contratado Doctor adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

D. **Carmen Paz Suárez Araujo**, con N.I.F. 43640373N profesora Catedrática de Universidad adscrita al Departamento de Informática y Sistemas de la Universidad de Las Palmas de Gran Canaria, como cotutora

C E R T I F I C A (N)

Que la presente memoria titulada:

“NeuroAutoTests: Aplicación web para el diagnóstico automático e inteligente del deterioro cognitivo”

ha sido realizada bajo su dirección por D. **Francisco Jesús Mendes Gómez**, con N.I.F. 42293367R

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 13 de septiembre de 2022

Agradecimientos

A mi familia, especialmente a mi padre, mi madre, mi hermana y Nancy por siempre estar apoyándome en todos los momentos duros y dándome fuerzas para seguir adelante, también por todo el sacrificio que hacen para brindarme la oportunidad de estudiar en la universidad una carrera que me apasiona y proveerme de un entorno lo más cómodo posible, dentro de nuestras posibilidades, para poder conseguirlo, todo lo que he logrado se lo debo a ellos, gracias por tanto.

A mis compañeros de trabajo Francisco y Adrián por ayudarme con las tecnologías de este proyecto, explicándome con mucha paciencia y dedicación mis dudas con respecto a las tecnologías, muchas gracias.

A mis tutores Patricio y Carmen Paz por todo su apoyo y su feedback a la hora del desarrollo de la aplicación, gracias a eso pude mejorar mucho el resultado final.

A Othello por siempre apoyarme en los momentos duros de la carrera y siempre estando allí para brindarme su ayuda y su compresión.

A todas aquellas personas que me he encontrado a lo largo de la carrera tanto profesores como alumnos que me han ensañado muchas cosas valiosas que me han llevado hasta aquí.

Licencia

Resumen

Las enfermedades neurodegenerativas es un término usado para abarcar aquellas que atacan principalmente el sistema nervioso provocando muerte celular y degeneración en el tejido nervioso, lo que ocasiona afecciones en las funciones locomotoras, el lenguaje, la memoria, el razonamiento y otras capacidades de las personas de una manera progresiva pudiendo terminar con la pérdida de autonomía [1]. Una de estas patologías son las Demencias, concretamente la Enfermedad de Alzheimer, demencia de más prevalencia. Esta neuropatología, no tiene biomarcador específico y no es curable. Su detección en una etapa temprana se convierte en una necesidad, ya que ayuda a que los tratamientos retrasen su avance, pero esto no es una tarea sencilla, ya que las pruebas para detectar esta afecciones como puede ser la resonancia magnética, entre otras, resultan muy costosas y difíciles de llevar a cabo. En especial cuando se quiere cribar a un número elevado de personas como pueden ser pacientes en un hospital geriátrico. Existen otros criterios diagnósticos, menos invasivos, más económicos y más fáciles de realizar que otros, como los test neurocognitivos.

Por esta razón, se ha desarrollado en este Trabajo de Fin Grado (TFG) la aplicación NeuroAutoTest que es una plataforma web para que los especialistas puedan realizar de una manera rápida y ágil test neurocognitivos a pacientes, almacenando los resultados para su posterior análisis. Esta aplicación cuenta con un modelo de inteligencia artificial capaz de realizar diagnósticos de deterioro cognitivo leve (DCL) (En inglés, Mild Cognitive Impairment o MCI) basados en los resultados de los distintos test y parámetros de los pacientes como su edad y años de educación. Facilitando a los médicos una herramienta que los ayude a detectar deterioro cognitivo en etapas tempranas. Esto permitirá proponer tratamientos tempranos, pudiendo llevar a mejorar la calidad de vida del paciente.

Palabras clave: diagnóstico automático, inteligencia artificial, neurociencias, test neurocognitivo, deterioro cognitivo leve, aplicación web, enfermedades neurodegenerativas

Abstract

Neurodegenerative diseases is a term used to cover those that mainly attack the nervous system causing cell death and degeneration in the nervous tissue, which causes affections in locomotor functions, language, memory, reasoning and other capacities of people of a progressive way that can end with the loss of autonomy [1]. One of these pathologies is Dementia, specifically Alzheimer's Disease, the most prevalent dementia. This neuropathology has no specific biomarker and is not curable. Its detection at an early stage becomes a necessity, since it helps treatments delay their progress, but this is not an easy task since the tests to detect these conditions, such as magnetic resonance imaging, among others, are very expensive and difficult to carry out. Especially, when you want to sift a large number of people, such as patients in a geriatric hospital. There are other diagnostic criteria, less invasive, cheaper and easier to perform than others, such as neurocognitive tests.

For this reason, the NeuroAutoTest application has been developed in this project, which is a web platform so that specialists can perform neurocognitive tests on patients quickly and agilely, storing the results for later analysis. This application has an artificial intelligence model capable of diagnosing Mild Cognitive Impairment (MCI) based on the results of the different tests and patient's parameters such as their age and years of education . Providing doctors with a tool that helps them to detect cognitive impairment in early stages. This will allow proposing early treatments, which may lead to improving the patient's quality of life.

Keywords: *automatic diagnosis, artificial intelligence, neurosciences, neurocognitive test, mild cognitive impairment, web application, neurodegenerative diseases*

Índice general

Índice de figuras

Índice de tablas

Capítulo 1
Introducción

1.1 Objetivos

El objetivo de este TFG es la creación de una aplicación web, que sea capaz de brindar a los usuarios una herramienta para realizar diagnósticos automáticos de deterioro cognitivo leve mediante una inteligencia artificial dando como entrada al modelo resultados de test neurocognitivos de un paciente y diferentes datos personales como lo son la edad del paciente y los años de estudio, brindando también la posibilidad de que dichos test neurocognitivos sean realizados desde la propia aplicación sin necesidad de utilizar métodos tradicionales como rellenar los formularios a lápiz y papel.

Objetivos específicos:

1. Cada usuario dispondrá de su propio perfil
2. Cada usuario tendrá una suite para gestionar sus pacientes pudiendo crear nuevos pacientes, actualizar los datos de un paciente en concreto o eliminar los datos de los pacientes
3. Se tendrá la posibilidad de que los usuarios puedan realizar a sus pacientes diferentes test neurocognitivos
4. Realizar diagnósticos de deterioro cognitivo leve sobre pacientes que hayan hecho test neurocognitivos mediante la aplicación de una manera sencilla, rápida y automática
5. Tener una vista con estadísticas de un paciente, mostrando gráficas de evaluaciones longitudinales tanto de los test realizados por un paciente como los diagnósticos realizados utilizando los modelos de inteligencia artificial entrenados para este fin, así como también, poder ver las respuestas de cada uno de los test realizados por un paciente en concreto

1.2 Justificación del proyecto

Las enfermedades neurodegenerativas afectan al cerebro donde existe un proceso progresivo de degeneración de este, y muerte neuronal que ocasiona a la persona afectada muchos problemas locomotores, del habla, memoria, el razonamiento, equilibrio, entre otras funciones realizadas por el cerebro. Cuando se agravan se derivan en una pérdida de autonomía, donde la persona se vuelve dependiente de otras para poder vivir, ya que deja de tener la capacidad de valerse por sí mismo.

De aquí la importancia del diagnóstico, ya que muchas de estas enfermedades solo presentan síntomas evidentes cuanto están en una fase avanzada lo que dificulta su tratamiento, pero un síntoma que alerta sobre su posible aparición futura es la existencia de un deterioro cognitivo. Los métodos para diagnosticar estas enfermedades van desde test neurocognitivos hasta las resonancias magnéticas. Los métodos de detección más fiables suelen ser muy costosos por lo que proponer soluciones de coste bajo se ha convertido casi en una necesidad para poder hacer cribas sobre poblaciones de individuos susceptibles a tener este tipo de enfermedades como pueden ser las personas mayores. De allí surge la idea de este TFG, donde se llevó a cabo la creación una aplicación web llamada NeuroAutoTest, que permite a sus usuarios (médicos especialistas o personal sanitario) realizar distintos test neurocognitivos a los pacientes y, utilizando diversos parámetros de dicho paciente y los resultados de los test, la aplicación es capaz de brindar un diagnóstico a través de un modelos de inteligencia artificial que toma como entradas los datos antes mencionados, de esta manera podemos cribar una población a un coste mucho más asequible que haciendo pruebas más exhaustivas y costosas como una tomografía o una resonancia magnética, y de una manera más ágil y rápida. De esta manera se puede ayudar a una detección de temprana de enfermedades como la Enfermedad de Alzheimer o la demencia en pro de mejorar la efectividad de los tratamientos y mejorar la calidad de vida de esos pacientes.

1.3 Estado del arte

Existen diversos estudios y Trabajos de Fin de Grado con una temática similar como es el caso de [2], realizado en la Universidad de Las Palmas de Gran Canarias donde se han estudiado muchos clasificadores basados en redes neuronales híbridas.

También tenemos la aplicación web Cogmultest [3] realizado por un TFG de la Universidad de La Laguna, donde se llevó a cabo la implementación de una aplicación web que ofrecía la posibilidad de realizar diferentes test neurocognitivos de manera digital o de manera más profesional la aplicación Neotiv [4] creada con el propósito de detectar la enfermedad de Alzheimer en una etapa temprana desarrollando nuevos test neurocognitivos que pueden ser fácilmente realizados desde la aplicación móvil. En el artículo [5] observamos que realizar los test de esta manera comparado a la manera tradicional de usar lápiz y papel también arroja buenos resultados con respecto a su acierto al diagnóstico con valores de especificidad y sensibilidad superiores al 0,80 en

algunos test.

Cabe a mencionar la gran cantidad de investigaciones y estudios cuyo tema central es el uso de inteligencia artificial para el diagnóstico automático de enfermedades neurodegenerativas, no solo a través del uso de los test neurocognitivos, sino también, analizando imágenes de resonancia magnéticas como el estudio [6].

Este campo hoy en día está muy estudiado, tanto por profesionales de la salud y las neurociencias como también profesionales del ámbito tecnológico e informático, trabajando en el desarrollo de herramientas que mejoren la calidad de vida de las personas que padecen este tipo de enfermedades. Posiblemente, en un futuro estas herramientas puedan ser determinantes en la cura y prevención de enfermedades neurodegenerativas como la Enfermedad de Alzheimer o la enfermedad del Parkinson.

1.4 Marco teórico

1.4.1 Test neurocognitivos

Un test neurocognitivo son un conjunto de pruebas diseñadas para evaluar si una persona tiene problemas de cognición, es decir, si tiene problemas en los procesos mentales que intervienen en los aspectos de la vida cotidiana, como por ejemplo la memoria o el habla, entre otras funciones del cerebro [7]. Como ya se ha comentado anteriormente, estos test se utilizan para detectar deterioro cognitivo leve, ya que a través de estos test podemos notar pequeñas variaciones en las funciones cognitivas de una persona que en su vida cotidiana pueden no ser perceptibles y que pueden ser el inicio de enfermedades más graves como la demencia cortical, Enfermedad de Alzheimer u otras clase de demencias.

Existen muchos test neurocognitivos, como por ejemplo los que han sido implementados en el caso de NeuroAutoTest y describimos a continuación:

1. Geriatric Depression Scale (GDS)

Este test neurocognitivo está enfocado al diagnóstico de la depresión, si bien este test no es capaz por si solo de diagnosticarla, si es capaz de arrojar pistas sobre su posible existencia, en su forma extendida es un test que cuenta con 30 preguntas de respuesta sí o no, pero en su forma corta se reducen a 15, donde 10 preguntas, al ser respondidas positivamente, indican presencia de síntomas de depresión, mientras que las 5 restantes lo indican si son respondidas negativamente. Cada pregunta se puntúa con un 1 si la respuesta indica depresión y con un 0 si no, al final los resultados de van desde el 0 al 15, donde las puntuaciones se interpretan de la siguiente manera:

- 0 - 4 se considera un puntuación normal

- 5 – 8 indica una depresión leve
- 9 – 11 indica una depresión moderada
- 12 – 15 indica una depresión severa

En [8] se puede encontrar más información sobre el test y como son las preguntas.

2. Functional Assessment Questionnaire (FAQ)

Es un test sencillo con 10 preguntas donde se interroga al paciente que tanta facilidad tiene para llevar a cabo labores de la vida cotidiana. En base a su respuesta ésta se puntúa en una escala del 0 al 3, la puntuación final será una suma de éstas estando en un rango de 0 al 30, donde una puntuación mayor o igual a 9 indica un deterioro funcional y un posible deterioro cognitivo, cada pregunta se puntúa según los siguientes criterios:

- 3 puntos si depende de otra persona para realizar las acciones
- 2 puntos si necesita ayuda para realizar la actividad
- 1 punto si se le dificulta pero es capaz de realizar la actividad solo o si nunca realizó la actividad pero podría suponer dificultad llevarla a cabo
- 0 puntos si es algo que normalmente hace sin problemas o si nunca realizó la actividad pero no tendría problemas en realizarla

En [9] se puede encontrar más información sobre el test y como son las preguntas.

3. Mini Mental State Evaluation (MMSE)

Esta prueba esta orientada a detectar deterioro cognitivo leve en los pacientes, contando con 30 preguntas que miden la orientación espacio temporal de paciente, la capacidad de atención, la capacidad de cálculo, la memoria, el lenguaje, la capacidad de seguir instrucciones y la percepción viso-espacial. Donde según la pregunta se pedirá al paciente que realice alguna acción que simplemente responda a una cuestión sencilla y esta será puntuada con un valor numérico si la respuesta o la acción ejecutada son correctas. Los resultados van en un rango de 0 a 30, la interpretación de los resultados es de la siguiente manera:

- 30 – 27 sin deterioro cognitivo
- 26 – 25 con posible deterioro cognitivo

- 24 – 10 con deterioro cognitivo leve a moderado
- 9 - 6 con deterioro cognitivo moderado a severo
- 6 - 0 con deterioro cognitivo severo

En [10] podemos ver un ejemplo del test y como llevarlo acabo.

1.4.2 Perceptrón multicapa (MLP – Multi Layer Perceptron)

Las redes neuronales artificiales (RNAs) o por sus siglas en inglés (ANN - Artifitial Neuronal Network) es un enfoque de computación no programada y un modelo de aprendizaje automático que son frecuentemente utilizadas en medicina para el diagnóstico automático. Uno de los modelos más conocidos de ellas es el Perceptrón multicapa. Antes de saber sobre este modelo miremos que es un perceptrón.

El **perceptrón** o **perceptrón simple** es una neurona artificial que define la unidad de las redes neuronales, donde son llevados a cabo aquellos cálculos para detectar las características o tendencias de los datos de entrada [11].

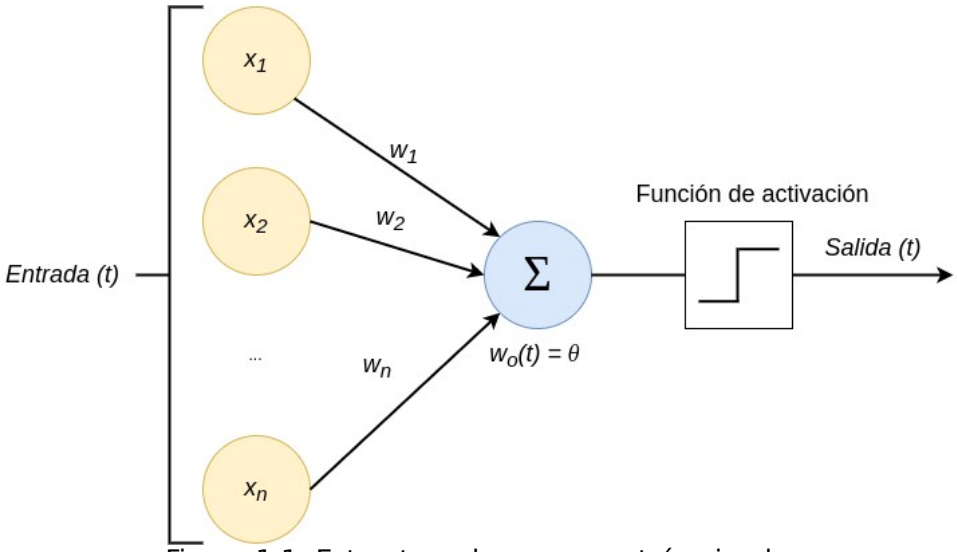

Figura 1.1: Estructura de un perceptrón simple

Básicamente esta neurona recibe un conjunto de señales de entrada (x_1, x_2, ..., x_n) los cuales multiplica por unos coeficientes de peso (w_1, w_2, ..., w_n) y los suma. Si el valor de esta suma supera un umbral la neurona se activará retornando un resultado, luego ese resultado predicho será comparado con el resultado conocido y el error calculado será propagado hacia atrás para poder reajustar los pesos (véase figura 1.1).

Con esta idea clara, el modelo **perceptrón multicapa** [12] es una red neuronal artificial que es capaz de resolver diferentes problemas como lo pueden ser el de reconocimiento de patrones o el que es de interés en este TFG, problemas de clasificación. Como su nombre indica este modelo está conformado por múltiples capas de neuronas clasificadas en 3 tipos:

Capa de entrada: la cual está conformada por todas aquellas unidades que reciben un patrón de entrada a la red y donde no se producen procesamientos de estas entradas.

Capas ocultas: que son las encargadas del procesamiento, formadas por neuronas donde la salida de una neurona es la entrada de las neuronas de la capa siguiente.

Capa de salida: es aquella donde las neuronas se corresponden con la salidas posibles de toda la red.

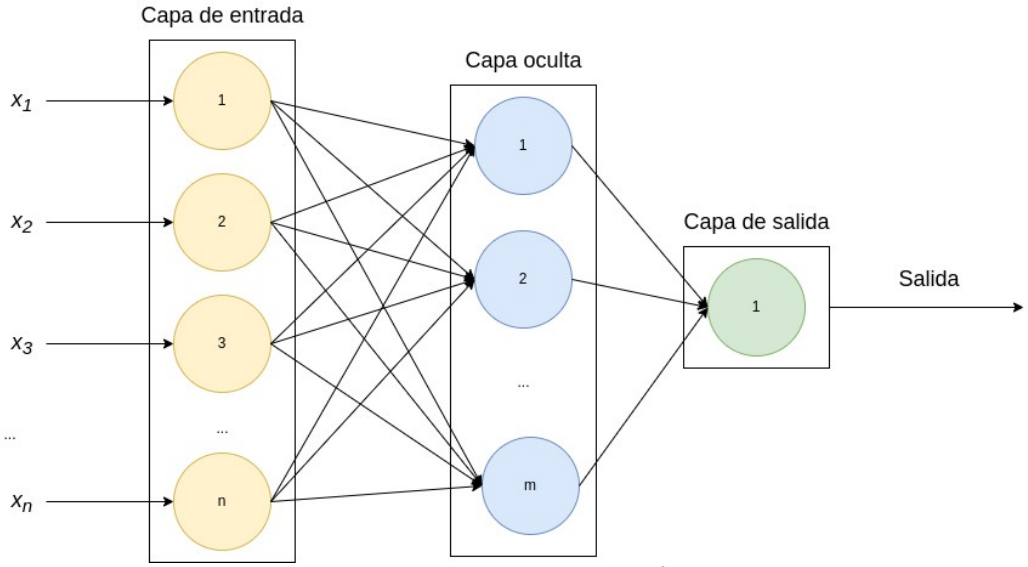

Figura 1.2: Estructura de perceptrón multicapa

Este modelo es capaz de aprender gracias al algoritmo de propagación hacia atrás o también conocido como backpropagation [13], este algoritmo básicamente se encarga de propagar error de la salida hacia las capas anteriores, alterando los pesos en las neuronas después de procesar la información. El cálculo del nuevo peso se realiza utilizando el descenso del gradiente [14]. Esta técnica de aprendizaje supervisado [15] consigue mejorar sus resultados aproximando sus salidas a los valores deseados y, por tanto, disminuyendo el error (véase figura 1.2).

Capítulo 2
Desarrollo de NeuroAutoTest

2.1 Descripción general de la aplicación

La aplicación, como se ha mencionado antes, es una plataforma para que los médicos puedan realizar a sus pacientes diferentes test neurocognitivos y poder realizar un diagnóstico automático basado en los resultados de estos y en datos como la edad y años de estudio del paciente. Dichos test pueden ser realizados de formar rápida, lo cual es útil para hacer cribados a muchas personas en poco tiempo y permiten realizar anotaciones en cada una de las preguntas que se realiza al paciente, que puede ser de interés a posteriori para que el médico pueda evaluar el caso con más detalle. Toda la aplicación está hecha de manera responsiva, por lo que puede ser usada en distintos dispositivos adaptándose al tamaño de la pantalla, lo que permite que no sea necesario un equipo incomodo para transportar y que solo baste con un portátil, tableta o el propio móvil. Aparte, permite que cada médico tenga un sistema de gestión de sus pacientes, donde podrá añadir nuevos, actualizar sus datos o eliminarlos de la base de datos, por último, los usuarios serán capaces de de visualizar gráficas sobre los diagnósticos, resultados y respuestas de los test que se hayan realizado anteriormente.

2.2 Herramientas utilizadas en el proyecto

En este capítulo hablaremos de todas la herramientas utilizadas en el desarrollo de la aplicación, tanto para el frontend, el backend y la base de datos. Empecemos hablando sobre los lenguajes de programación.

- **Lenguajes de programación utilizados en la aplicación:**

JavaScript: Utilizado en el desarrollo del frontend de la aplicación junto al framework React. Este es uno de los lenguajes más populares para el desarrollo web, y cuenta con una gran cantidad de librerías [16].

Python 3: En cuanto al backend de a aplicación este está construido en Python, básicamente la elección de este lenguaje para el desarrollo de la

aplicación viene fundamentada en que, mayormente, las herramientas para trabajar con modelos de inteligencia artificial están construidas para ser usadas en este lenguaje, además Python es un lenguaje interpretado bastante sencillo de utilizar pero muy potente teniendo muchas librerías creadas y que permite que puedas hacer casi cualquier cosa con él con muy pocas líneas de código [17].

Las herramientas usadas en el desarrollo de la aplicación, más a allá de los lenguajes de programación, son:

- **Frontend:**

React: Como dice en su página web oficial [18] React es una librería para construir interfaces gráficas de usuario de una manera sencilla que trabaja con JavaScript, lo cuál, también nos brinda la capacidad de crear diferentes componentes y vistas que serán renderizadas en nuestro navegador en función del estado de la aplicación. Debido a esto se ha convertido en una de las herramientas líder en el mercado del desarrollo web. Por estos motivos se decidió usar en esta aplicación.

Auth0: Es una plataforma para la gestión y autentificación de usuarios, lo cuál hace que la autentificación de nuestras aplicaciones sea sencilla de manejar, ya que es fácil de instalar en nuestra aplicación y, a través del portal web, puedes controlar todo lo relacionado a tus usuarios. Tiene una buena integración con React lo cuál la hacía perfecta para este proyecto [19]. A pesar de ser un servicio de pago cuenta con un plan gratuito bastante completo.

- **Backend**

Flask: Esta herramienta es un framework de Python [20] para el desarrollo de aplicaciones web. En el cuál podemos crear una API (Aplication Programming Interface) REST (Representational State Transfer) [21] de manera sencilla. Permitiendo hacer desde nuestro frontend peticiones HTTP (Hypertext Transfer Protocol) [22] y, poder tener toda la lógica de nuestra aplicación y la conexión a las bases de datos del lado del servidor.

Nginx: Es un servidor web de código abierto, el cuál es capaz de servir contenido mediante peticiones HTTP, usada en el caso de la aplicación como un Content Delivery Network (CDN) [23] para servir el frontend de nuestra aplicación a al navegador a través de peticiones HTTP.

- **Base de datos**

PostgreSQL: Es un sistema gestor de bases de datos de código abierto basado en el modelo relacional [24] y uno de los más usados a nivel mundial [25], funciona a través del lenguaje de consultas SQL (Structured Query Language) [26] el cuál nos permite hacer consultas para obtener los datos que queremos desde el servidor de base de datos. Usado en este TFG como base de datos principal de la aplicación.

- **Modelo diagnosticador**

Cuadernos Jupyter: Usados para entrenar los modelos en conjunto con Google Colab [27] el cuál nos permite ejecutar código en Python de una manera segmentada, permitiendo ejecutar pequeños trozos de código y si algo falla no tener que ejecutar todo desde el principio [28], esto último es especialmente útil en campos como el aprendizaje automático o la ciencia de datos a la hora de realizar experimentaciones.

Scikit-learn: Es una librería de Python para aprendizaje automático que cuenta con una amplia gama de modelos de inteligencia artificial [29], y que fue usada en este TFG para entrenar y usar los modelos diagnosticadores.

- **Despliegue**

Docker: La aplicación es desplegada de manera local gracias a Docker. Docker es un proyecto de código abierto que permite ejecutar procesos en un entorno aislado dentro del sistema operativo lo que se conoce como contenedor. Estos contenedores permiten que el proceso pueda ser ejecutado en diferentes entornos ya que no dependen del sistema operativo en el que usamos Docker. Al crear un contenedor creamos lo que se llama una imagen, que es básicamente la definición de nuestro contenedor, lo cuál nos permite compartirlas una vez construidas y pudiendo ser usadas para crear imágenes mas complejas añadiéndole capas, de esta manera podemos crear contenedores más complejos capaces de ejecutar una aplicación entera en cualquier entorno [30].

2.3 Infraestructura de la aplicación

En la figura 2.1 podemos observar un diagrama con la arquitectura de la aplicación.

Primero, los usuarios desde el navegador, al momento de iniciar sesión, son redirigidos al servicio de autentificación de Auth0 para tener su ID de usuario, una vez obtenido, son redirigidos nuevamente a la aplicación de NeuroAutoTest, donde ya para este momento estamos autenticados.

El frontend, servido por el servidor web de Nginx, a través de peticiones HTTP realiza llamadas a la API REST creada con Flask en nuestro backend para obtener los datos necesarios para las diferentes vistas de la aplicación, actualizar y cargar datos en PostgreSQL o eliminarlos a través de estas llamadas a la API.

En cuanto al backend, este utiliza el conector de psycog2 [31] (una librería de Python) para realizar las consultas necesarias a la base de datos, así como la librería Scikit-learn para cagar los modelos previamente entrenados y obtener los diagnósticos dados por dichos modelos.

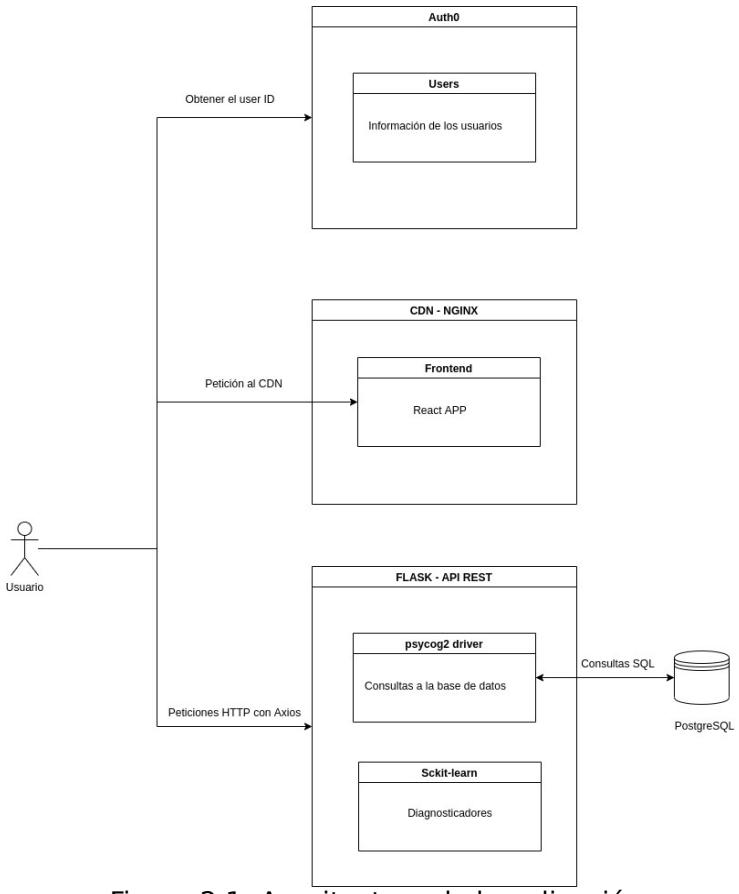

Figura 2.1: Arquitectura de la aplicación

2.3.1 Base de datos

Como se ha comentado antes para este proyecto, se decidido utilizar PostgreSQL como gestor de bases de datos donde gracias a Docker levantamos un servidor con el gestor de bases de datos con la dirección http://localhost:8000. A la hora de diseñar la base de datos se han tomado en cuenta los siguientes casos de uso:

- Cada usuario (Médico o especialista de la salud) puede tener múltiples pacientes
- Cada paciente puede tener diferentes resultados de los test neurocognitivos realizados
- Cada paciente puede tener diferentes diagnósticos realizados con los modelos de inteligencia artificial.

Para este fin se ha creado un esquema en la base de datos llamado “neuroautotest” donde estarán todas las tablas necesarias para la aplicación, dentro de este esquema encontraremos las siguientes tablas:

- **USERS**: Cada fila de la tabla representa a un usuario (Médico) de la aplicación, esta compuestas por 5 columnas:
  - "*user_id*": que es el ID de usuario de Auth0 usado para identificar de manera inequívoca a los usuarios y el la clave primaria de la tabla
  - "*birth_date*": con la fecha de nacimiento del usuario
  - "*first_name*": con el nombre del usuario
  - "l*ast_name*": con el apellido del usuario
  - "*email*": con la dirección de correo electrónico del usuario, la cuál, también es el nombre de usuario que utiliza Auth0 para autentificar a los usuarios

- **PATIENTS**: Cada fila de esta tabla representa un paciente, aquí encontraremos todos los datos de los pacientes, en esta tabla contamos con las columnas:
  - "*patient_id*": como clave primaria y que contiene un identificador único para cada paciente
  - "*user_id*": como clave extrajera. Como comentamos antes, es el ID de usuario de los usuarios de la aplicación, gracias a este campo establecemos la relación entre un paciente y un médico o usuario
  - "*birth_date*": con la fecha de nacimiento del paciente
  - "*first_name*": con el nombre del paciente
  - "*middle_name*": con el segundo nombre del paciente
  - "*last_name*": con el apellido del paciente
  - "*observations*": con observaciones y datos adicionales que el usuario quiera destacar sobre los pacientes
  - "*years_of_studies*": que representa los años de estudio que tiene un paciente
- **RESULTS:** En esta tabla almacenamos los datos relativos a los resultados de los test neurocognitivos realizados por los pacientes, contando con las siguientes 5 columnas:
  - "*patient_id*": con de identificador del paciente que ha hecho test, esta columna es una clave extranjera que establece la relación entre un paciente y un resultado
  - "*test_id*": con el identificador del test realizado
  - "*test_date*": con la fecha en la que se realizó el test

Estas tres columnas representan una clave primaria compuesta para identificar inequívocamente un resultado.

- “*anwers*”: almacena una cadena de caracteres que representa toda la información de las respuestas del paciente al test realizado
- “*result*”: puntuación del test

- **DIAGNOSTICS:** En esta tabla almacenamos los diagnósticos realizados con los modelos de inteligencia artificial, contando con las siguientes 5 columnas:
  - “patient_id”: con de identificador del paciente que ha hecho test, esta columna es una clave extranjera que establece la relación entre un paciente y un diagnóstico
  - “model_id”: con los datos de entrada que utilizó el modelo para realizar el diagnóstico del test realizado
  - “diagnostic_date”: con la fecha en la que se realizó el test

Estas tres columnas representan una clave primaria compuesta para identificar inequívocamente un resultado.

- “reliability”: es la probabilidad con la que el modelos de inteligencia artificial realiza el diagnóstico
- “result”: resultados del diagnóstico, es un dato booleano Si/NO

Veamos todo de una manera más clara en la figura 2.2.

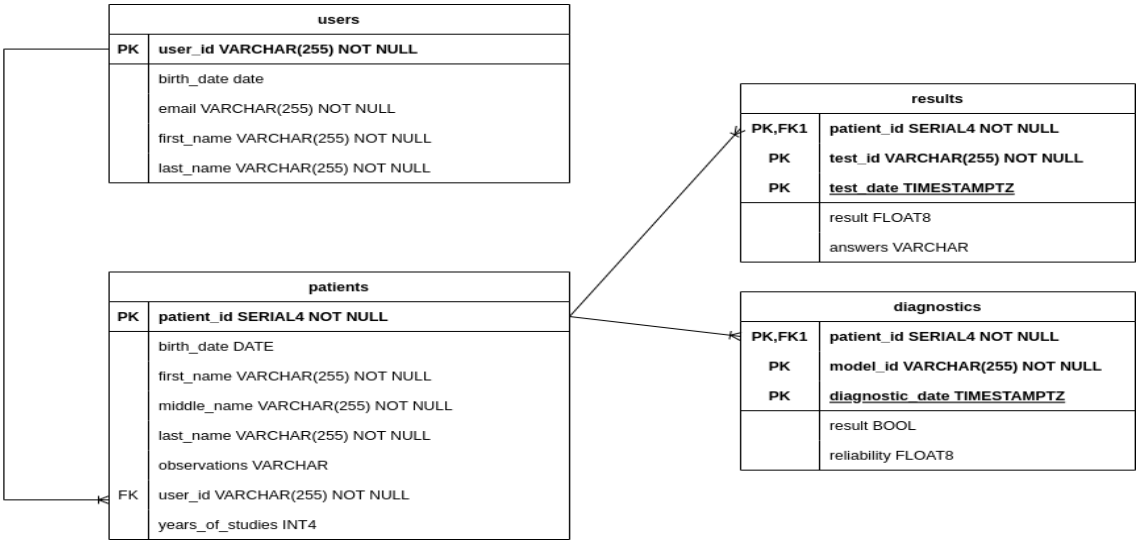

Figura 2.2: Modelo Entidad-Relación de la base de datos de NeuroAutoTest

El diseño pese a ser un diseño simple permite escalarlo en función de las nuevas funcionalidades y necesidades que tenga la aplicación.

2.3.2 Frontend

Como se ha comentado antes, el frontend está creado con React y JavaScript para empezar el proyecto se ha utilizado el comando de Node.js [32]:

- $ npx create-react-app frontend

Este comando crea de manera automática en el directorio frontend toda la estructura básica de la aplicación en React donde podemos empezar a crear los diferentes componentes de la interfaz gráfica de usuarios, también cabe a destacar la librería Axios [33], una librería de JavaScript, que proporciona un cliente HTTP para que de manera programática podamos hacer peticiones a nuestro servidor backend. Con esta librería se ha creado una API cliente para el realizar llamadas a la API REST de nuestro backend y, que estas puedan ser ejecutadas desde cualquier parte del código sin necesidad de volver a escribir las mismas peticiones HTTP. Otra parte fundamental es Auth0 que es nuestro gestor de autentificación y de usuarios. Auth0 ya provee módulo para React con diferentes funcionalidades para configurar y obtener la información de los usuarios y redirigir a la página de inicio de sesión que también nos proporciona Auth0 (véase figura 2.3).

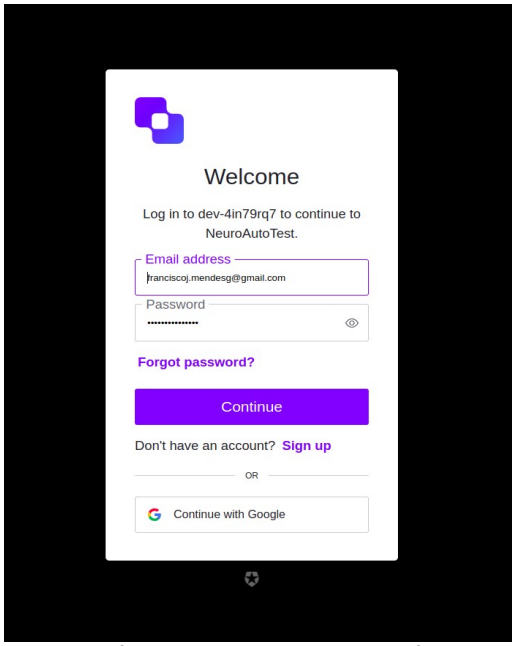

Figura 2.3: Página de inicio de sesión de Auth0

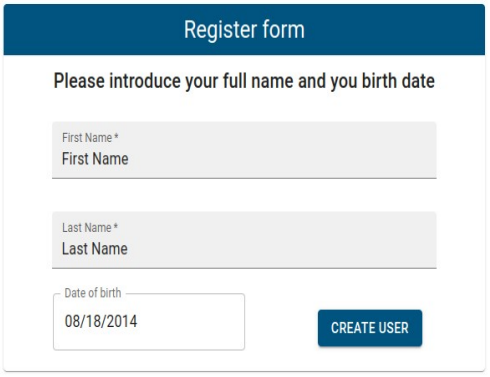

Figura 2.4: Registro de NeuroAutoTest

Después de obtener el ID del usuario desde Auth0, este coincidirá con el ID del usuario de nuestra base de datos, con él tendremos acceso exclusivamente

a los datos que le pertenecen a ese usuario. La primera vez que inicie sesión en la aplicación, una vez dados de alta en Auth0 y obteniendo el ID, aparecerá un formulario de registro (véase figura 2.4) para rellenar los datos del usuario que introduciremos en la base de datos como el nombre, la fecha de nacimiento, el ID de Auth0 y el correo electrónico.

Los directorios del frontend están organizados de la siguiente manera:

- "*public/*" : este directorio contiene los ficheros estáticos de React y el fichero "index.html" donde se inyectará nuestra aplicación de React
- "*src/config*": que contiene diferentes ficheros de configuración de la aplicación, entre ellos, el fichero con la configuración de Auth0 para poder conectarnos con el servicio
- "*src/components*": contiene todos los componentes creados y utilizados en cada una de las vistas
- "*src/views*": contiene cada una de las vistas de la aplicación.
- "*src/layouts*": contiene los componentes principales de la aplicación como la cabecera y el cuerpo donde aparecerán las vistas de la aplicación
- "*src/client*": Contiene la API cliente para hacer las consultas al Backend
- "*index.js*": fichero con el punto de entrada de la aplicación

Mencionar que la mayoría de los componentes de la aplicación esta creado con un librería de componentes para React llamada Material-UI [34], la cuál fue especialmente útil ,ya que puedes crear tus propios componentes a partir de platillas de una manera más rápida y ágil. También te permite personalizar los componentes tanto como quieras haciendo modificaciones en el CSS o directamente desde distintas APIs que te proporciona la propia librería de Material-UI.

2.3.3 Backend

En el apartado 2.2 de este capítulo pudimos observar que el backend esta desarrollado usando Python y Flask para crear una API REST que devolverá la información necesaria al frontend a través de peticiones HTTP. La decisión de usar estas tecnologías es debido a uso de la librería Scikit-learn de Python, para así poder trabajar con los modelos de inteligencia artificial que hemos entrenado. La API REST tiene una serie de "*Endpoints*" divididos en dos categorías:

1. Endpoints para solicitar datos a PostgreSQL:

- Endpoint “/users/{user_id}” recibe la variable user_id que será la ID del usuario en Auth0
    - Petición GET: Retorna los datos de un paciente almacenado en PostgreSQL con una respuesta HTTP response 200 OK
- Endpoint “/users/insertUser”:
    - Petición POST: Recibe un JSON (JavaScript Object Notation) [35] con los datos del usuario que se desea insertar, si estos datos son correctos, devuelve un HTTP response 200 OK con un mensaje JSON personalizado.
- Endpoint “/patients/getPatients”:
    - Petición POST: Recibe un JSON con los datos de un usuario y retornará un JSON con todos los datos de todos los pacientes que posee ese usuario y un HTTP response 200 OK
- Endpoint “/patients/createPatient”:
    - Petición POST: Recibe un JSON con los datos de un paciente para crearlo en la base de datos. Retornará un mensaje JSON personalizado y un HTTP response 200 OK
- Endpoint “/patients/updatePatient”:
    - Petición POST: Recibe un JSON con los datos de un paciente para actualizar sus datos y retornará un mensaje JSON personalizado y un HTTP response 200 OK
- Endpoint “/patients/deletePatient”:
    - Petición POST: Recibe un JSON con el ID de un paciente y lo elimina de la base de datos, retornando un mensaje JSON personalizado y un HTTP response 200 OK
- Endpoint “/test/saveResults”:
    - Petición POST: Recibe un JSON con los resultados de un test neurocognitivo realizado a un paciente y lo almacena en la base de datos. Retorna un mensaje JSON personalizado y un HTTP response 200 OK
- Endpoint “/test/lastTestResult”:
    - Petición POST: Recibe un JSON con el ID de un paciente y un tipo de test para buscar en la base datos el último resultado de ese tipo test para un paciente. Retorna un JSON con los datos de del resultado de ese test y un HTTP response 200 OK
- Endpoint “/test/allTests”:
    - Petición POST: Recibe un JSON con el ID de un paciente, retornando

un JSON con todos los resultados de los test realizados por ese paciente y un HTTP response 200 OK

- Endpoint “/test/AllTestScores”:
  - Petición POST: Recibe un JSON con el ID de un paciente y un tipo de test para buscar en la base de datos los resultados de todos las puntuaciones de un tipo de test para un paciente concreto, retornando todas esta puntuaciones en un objeto JSON y un HTTP response 200 OK
- Endpoint “/AllDiagnostics”:
  - Petición POST: Recibe un JSON con el ID de un paciente y un tipo de modelo de Inteligencia para buscar en la base de datos los resultados de todos los diagnósticos de un paciente realizados por un diagnosticador en concreto, retornando todos estos diagnósticos en un objeto JSON y un HTTP response 200 OK

2. Endpoints para solicitar inferencias:

- Endpoint “/inference”:
  - Petición POST: Recibe un JSON con las entradas necesarias para realizar una inferencia utilizando un modelo diagnosticador y el tipo de modelo a usar, luego se procesa la petición realizando el diagnóstico, se retorna un HTTP response 200 OK y un objeto JSON con la predicción del modelo, y la probabilidad de la predicción, almacenado automáticamente este diagnóstico en la base de datos

La estructura de directorios para el caso del backend es bastante sencilla y esta creada de esta manera ya que es la que recomienda Flask para los proyectos que lo usan:

- “*app/static/ia-models/MLP*”: contiene todos los modelos entrenados en un formato “.PKL” [36]
- “*app/app.py*”: contiene la definición de todos los Endpoints, la lógica del servidor y la API REST con Flask. Es el programa principal
- “*app/database_connector.py*”: contiene todas las funciones y consultas necesarias para obtener datos desde PostgreSQL
- “*app/ai_models.py*”: contiene las funciones para cargar los modelos entrenados y para realizar inferencias con los distintos modelos

Capítulo 3
Resultados de la aplicación

3.1 Cuerpo de datos: repositorio ADNI

A la hora del entrenamiento del modelo se utilizó información procedente del repositorio de ADNI (Alzheimer's disease Neuroimaging Initiative) [37], el cuál contiene una gran cantidad de datos de estudios longitudinales relacionados con el Alzheimer y su detección. Esta iniciativa tiene como principal objetivo la detección temprana de la Enfermedad de Alzheimer y encontrar maneras de poder hacer seguimiento de esta. Estos datos ya han sido filtrados y podemos observar un resumen de los mismos en la tabla 3.1. Donde podemos apreciar el grupo de control (sin deterioro cognitivo leve) y los pacientes que poseen un deterioro cognitivo leve, así como las métricas de sus datos personales y resultados en los diferentes test neurocognitivos.

	Control	**DCL**	**Total**
Sujetos	203	128	331
Edad media (DE)	74,1 (6,3)	74,9 (7,2)	74,4 (6,7)
Rango Edad	56,3 – 89,1	56,3-88,0	56,3-89,1
Años de estudio media (DE)	16,5 (2,6)	15,5 (3,2)	16,1 (2,9)
Rango Años de estudio	10-20	4-20	4-20
MMSE media (DE)	29,1 (1,2)	27,2 (1,7)	28,3 (1,7)
Rango MMSE	24-30	24-30	24-30
FAQ media (DE)	0,2 (0,6)	3,6 (4,4)	1,5 (3,3)
Rango FAQ	0-5	0-20	0-20
GDS media (DE)	0,8 (1,2)	1,6 (1,5)	1,1 (1,3)
Rango GDS	0-6	0-5	0-6

Tabla 3.1: Resumen de los datos de ADNI usados para el entrenamiento de los modelos.

DE: Desviación típica

Estos datos han sido partidos en 2 grupos, uno de entrenamiento con 255 personas y 76 personas para la validación, donde se ha tenido en cuenta la proporción entre personas con y sin DCL para mantener ese balance en ambos grupos (entrenamiento y validación).

Mencionar que estos datos filtrados han sido procesados por los autores del artículo [38] y facilitados por el tutor de este TFG Patricio García Báez.

3.2 Modelos entrenados

Los modelos utilizados son perceptrones multicapa entrenados con los datos mencionados anteriormente, en este caso se ha decidido entrenar y ajustar solo 5 modelos basados en los resultados del estudio [38], cuyos resultados podemos ver en la tabla 3.2.

Entradas	AUC	Precisión (%)	Sensibilidad	Especificidad
Edad + MMSE	0,82	76,32	0,91	0,53
Edad + GDS	0,55	65,79	**0,93**	0,23
Edad + FAQ	0,91	**86,84**	0,89	**0,83**
MMSE + GDS + Años de estudio	0,61	65,79	**0,93**	0,23
Edad + FAQ + MMSE	**0,94**	84,21	0,89	0,77

Tabla 3.2: Resultados de los modelos entrenados

La elección de estos modelos esta basado en el artículo [37] tomando los mejores diagnosticadores que tengan al menos un test de todas las posibles combinaciones, es decir, si tenemos solo datos para el test FAQ y el GDS utilizar el mejor modelo que necesite al menos uno de estos test. Los datos de la “Edad” y “Años de estudio” de los pacientes se supone que siempre los tendremos a la hora de realizar los diagnósticos. Basándonos en el AUC (Area Under the Curve) [39] podemos concluir que el mejor modelo entrenado es el Edad + FAQ + MMSE aunque no sea el que tiene la mayor precisión. Por lo que, para obtener diagnósticos en la aplicación siempre se recomendará al usuario que los pacientes realicen el test FAQ y el MMSE para una mayor fiabilidad en el diagnóstico. A pesar de que los estudios del artículo [37] han sido la base para seleccionar los modelos anteriores, los resultados de ambos estudios difieren entre sí, principalmente debido a que se usan distintos tipos de RNAs.

3.3 Casos de uso

La aplicación cuenta con varias vistas para realizar diferentes tareas, así como también un menú para la navegación y un sistema de autenticación basado en Auth0.

3.3.1 Menú y cabecera

En la cabecera de la aplicación podemos encontrar el menú de la aplicación con las pestañas de pacientes, test, evaluar y estadísticas (véase figura 3.1), además del avatar del usuario que ha iniciado sesión.

Si pulsamos este avatar, aparecerá un menú desplegable donde podemos acceder al perfil del usuario y el botón de cerrar sesión (véase figura 3.3).

La aplicación también, como se comentó anteriormente, es responsiva para que pueda ser usada desde dispositivos con pantallas más pequeñas como los móviles. En la figura 3.2 podemos observar como cambia la aplicación en un Iphone 12, se aprecia como ahora tenemos un botón con un menú desplegable con las diferentes vistas de la aplicación.

Figura 3.1: Cabecera de la aplicación

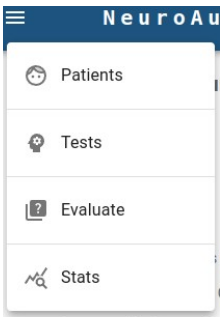

Figura 3.2: Menú desplegable

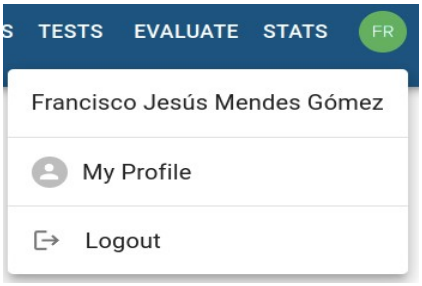

Figura 3.3: Menú personal y botón de cierre de sesión

3.3.2 Perfil

En la figura 3.4 observamos como en la aplicación se muestra la vista del perfil con los datos y el avatar de usuario.

Figura 3.4: Vista del perfil del usuario

3.3.3 Inicio de sesión

Lo primero que vemos al entrar a la aplicación es un mensaje de bienvenida (figura 3.5), y si pulsamos el botón de inicio de sesión, la aplicación nos redirige al servicio de autentificación de Auth0 (figura 3.6). Una vez autenticados, pasamos a la aplicación.

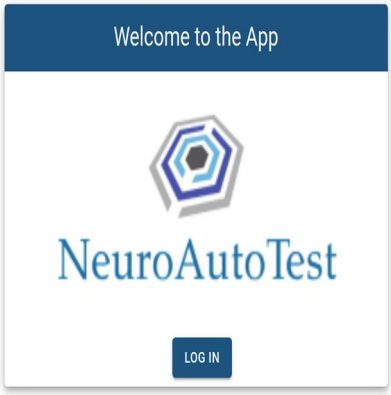

Figura 3.5: Bienvenida de la aplicación

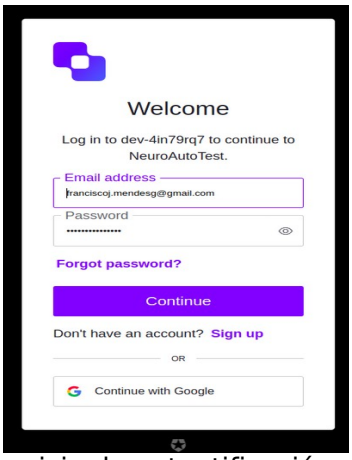

Figura 3.6: Servicio de autentificación de Auth0

3.3.4 Registro

Si el usuario es primera vez que entra a la aplicación después de haberse autenticado con Auth0, deberá registrar ciertos datos en la aplicación por lo que aparecerá el formulario de registro (figura 3.7), una vez completado se accederá a la aplicación. Este formulario no volverá a aparecer, solo lo hará la primera ve que entra un usuario a la aplicación.

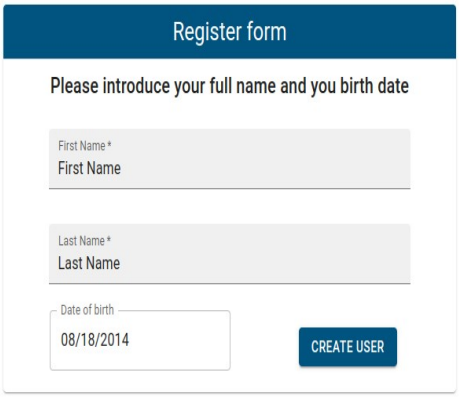

Figura 3.7: Formulario de registro

3.3.5 Gestión de pacientes

En la vista de gestión de pacientes (figura 3.8) encontraremos una tabla con los pacientes. Seleccionando un paciente, a la izquierda veremos los detalles del paciente seleccionado, también encontraremos 3 botones:

- Añadir paciente
- Actualizar paciente
- Eliminar pacientes

Al pulsar los botones nos aparecerá un formulario basado en la acción que deseemos realizar, como podemos observar en las figuras 3.9, 3.10 y 3.11.

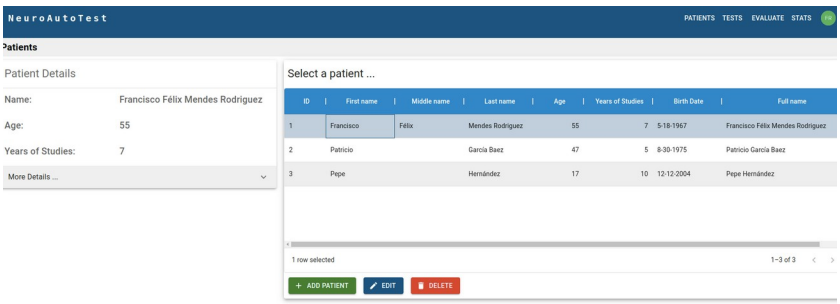

Figura 3.8: Vista de gestión de pacientes

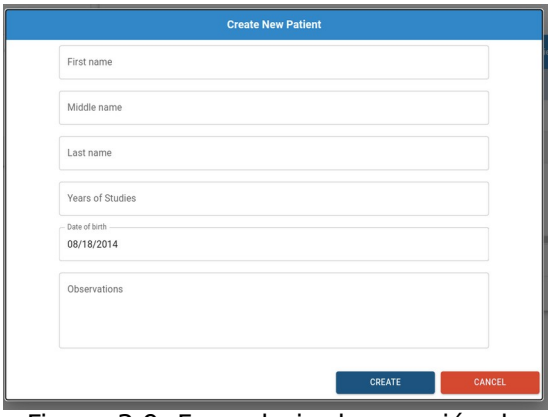

Figura 3.9: Formulario de creación de paciente

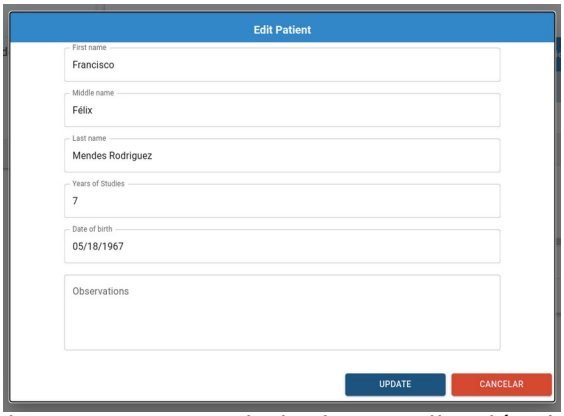

Figura 3.10: Formulario de actualización de datos de paciente

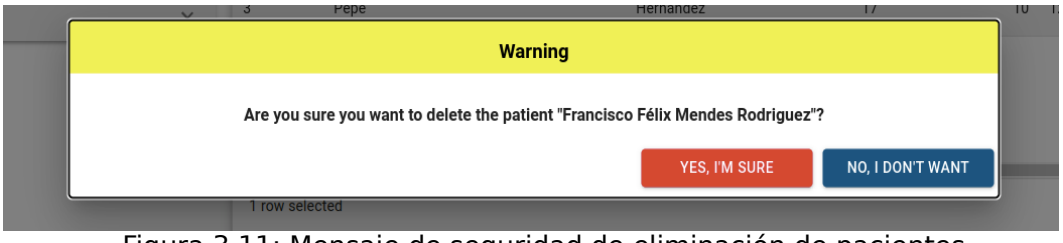

Figura 3.11: Mensaje de seguridad de eliminación de pacientes

3.3.6 Test neurocognitivos

Esta vista nos permite ver y seleccionar los test neurocognitivos con los que se quieran evaluar a los pacientes.

Seleccionaremos el paciente con el que queremos realizar un test, y luego pulsando sobre el botón “SELECT PATIENT” avanzaremos a la siguiente vista (véase figura 3.12).

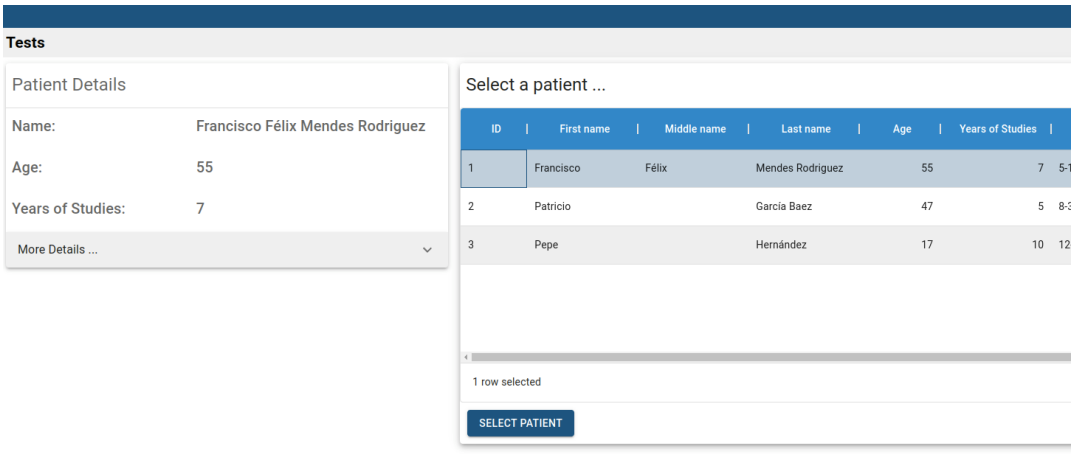

Figura 3.12: Selección de paciente en la vista de test

A continuación, encontraremos diferentes componentes: los detalles del paciente seleccionado, un acordeón con los diferentes test y un botón para volver a la selección de pacientes (véase figura 3.13).

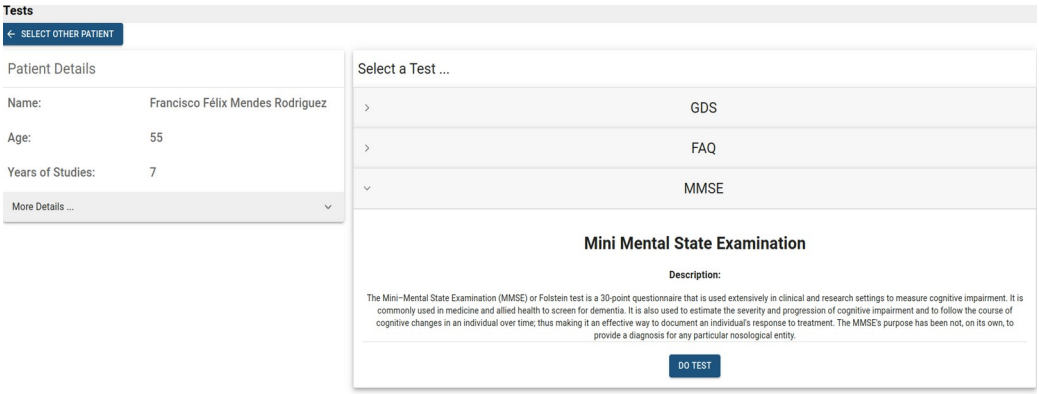

Figura 3.13: Selección de test

Como observamos en la figura 3.13, si seleccionamos un test se despliega un pequeño mensaje con la descripción del test y un botón para empezar el test. Pulsando en él podemos acceder directamente al test y poder rellenarlo, como podemos ver en las figuras 3.14 y 3.15.

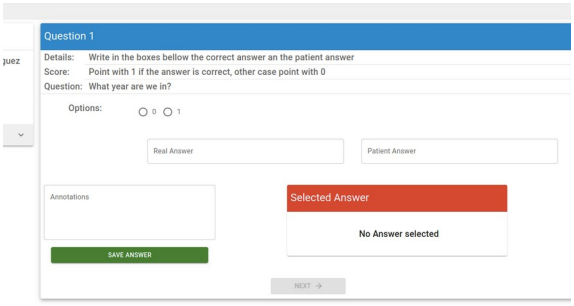

Figura 3.14: MMSE test primera pregunta (1)

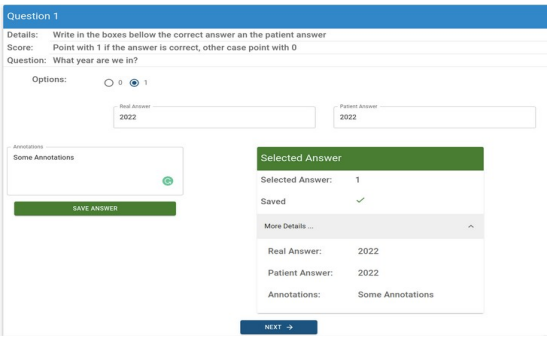

Figura 3.15: MMSE test primera pregunta (2)

Como observamos en la figura 3.16 las pregunta se componen por varios elementos: la pregunta en cuestión, un formulario para responder la pregunta, una caja para anotaciones que desee hacer el usuario, un botón para guardar la respuesta y un visualizador para ver los detalles de la respuesta dada y su estado (si está guardada o no). También encontramos botones para avanzar a la siguiente pregunta, volver a la pregunta anterior y para finalizar el test, los cuales irán apareciendo en función de la pregunta que estemos respondiendo en ese momento.

En la aplicación existen diferentes tipos de preguntas, como podemos observar a continuación:

- Preguntas con respuesta binaria: Son preguntas cuyas respuestas solo pueden ser “YES” o “NO”. Cada respuesta es puntuada de manera interna en función de cuál sea la correcta.

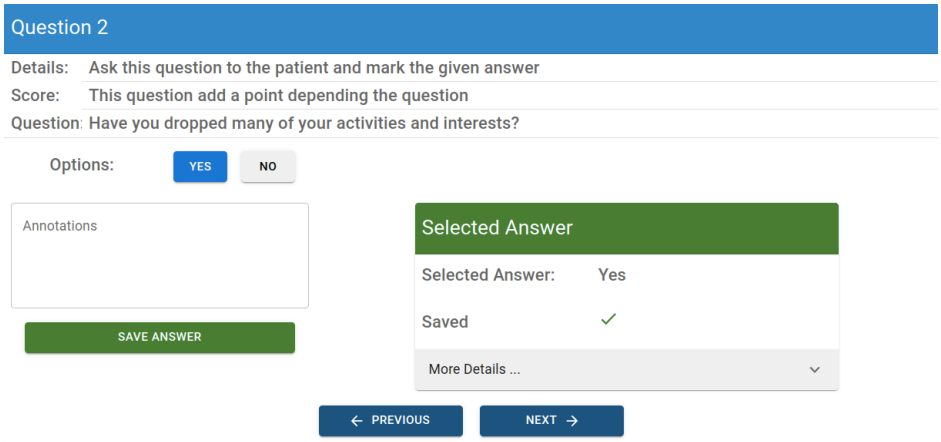

Figura 3.16: Pregunta con respuesta binaria

- Preguntas de evaluación numérica: Estas preguntas son puntuadas por el evaluador con una puntuación numérica en un rango que varía según la pregunta. Cada pregunta contiene detalles de como se debe puntuar las respuestas del paciente. (véase figura 3.18).

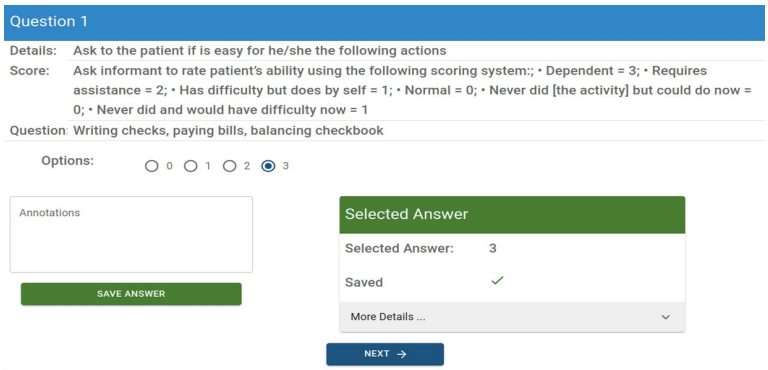

Figura 3.17: Preguntas de evaluación numérica

- Preguntas de evaluación numérica con respuestas: Similares a las preguntas anteriores, con la diferencia de que se pueden guardar dos valores, la respuesta real a la pregunta y la respuesta dada por el paciente. Este tipo de pregunta tiene dos formatos:
  - Formato libre: En este formato podemos escribir la respuesta correcta y la respuesta del paciente (véase figura 3.16). Las preguntas requieren que el paciente proporcione una respuesta que puede variar dependiendo del contexto en el que se esté realizando en test neurocognitivo. Por ejemplo, si la pregunta es: ¿Dónde estamos?, la respuesta correcta va a cambiar en función de la ubicación donde estemos realizando la prueba.
  - Formato selección: Este formato es parecido al anterior, con la diferencia de que ahora tenemos un conjunto de respuestas válidas acotado (véase figura 3.19). Por ejemplo, si la pregunta es: ¿Qué día de la semana es hoy?, ya sabemos de antemano todas la posibilidades que pueden ser respuestas correctas.

La puntuación asignada a este tipo de preguntas puede variar según la pregunta. La forma de puntuar estará especificada directamente en la pregunta.

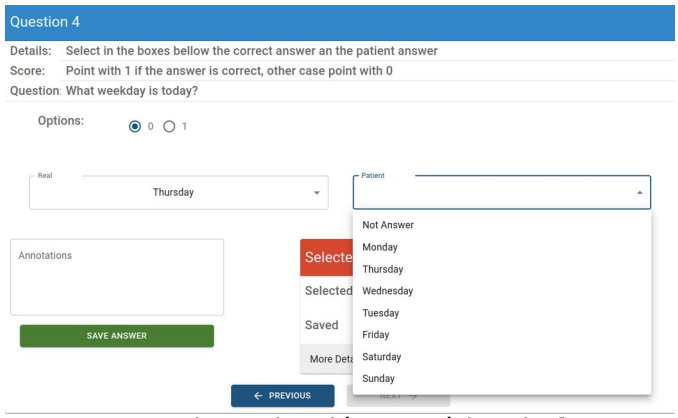

Figura 3.18: Pregunta de evaluación numérica de formato selección

- Preguntas de evaluación numérica con información añadida: Este tipo de preguntas son similares a las anteriores, pero, en este caso se requiere

que el evaluador rellene un cuadro de texto con información relevante a la pregunta. Esta información y la forma de puntuar la respuesta, estará especificada en la propia pregunta. En la figura 3.20 observamos que se pedirá al paciente que nombre tres objetos en intervalos de 1 segundo. En el cuadro de texto el evaluador escribirá dichas palabras separadas por comas, y puntuará la respuesta del paciente como se especifica en el apartado “Score” de la pregunta.

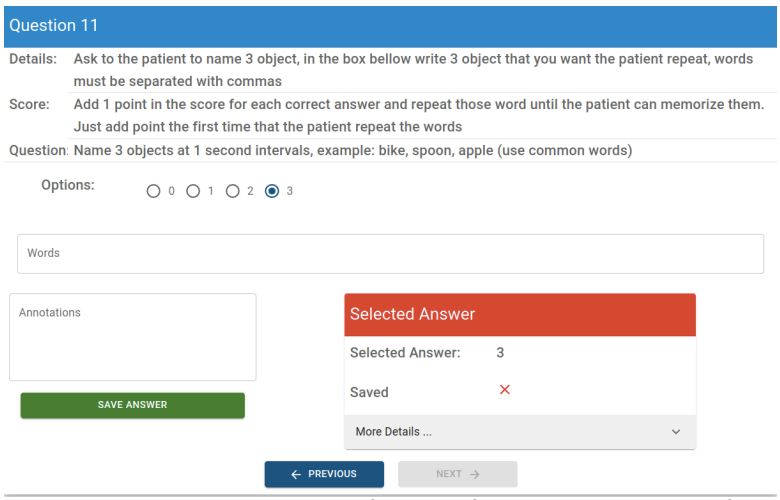

Figura 3.19: Pregunta de evaluación numérica con información añadida

- Preguntas opcionales: Este tipo de pregunta, realmente son dos preguntas, donde el paciente tendrá que responder ambas y el evaluador puntuará las repuestas de cada pregunta como corresponda. La aplicación, de manera automática, a la hora del cálculo de la puntuación final del test, elegirá la respuesta con mayor puntuación (véase figura 3.20).

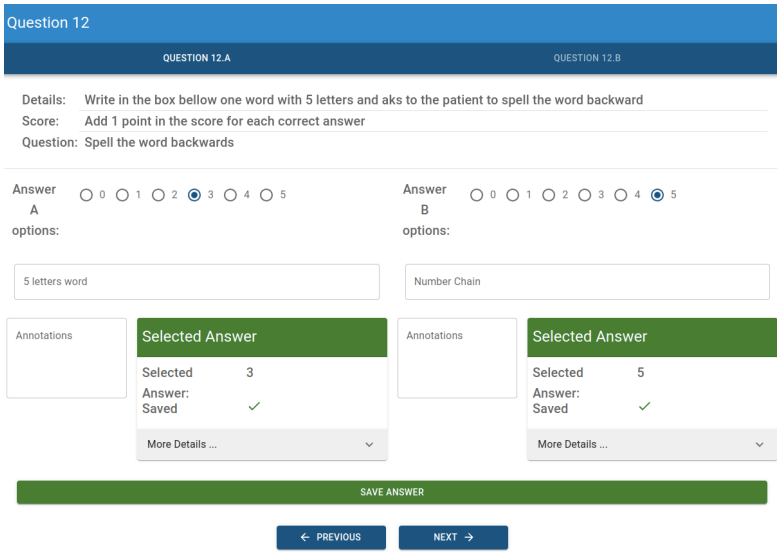

Figura 3.20: Pregunta opcional

- Realizar acción: En este tipo de preguntas, se necesita que el paciente lea una acción y la realice. En la figura 3.21 vemos un ejemplo. Aquí el evaluador escribirá una acción en el cuadro de texto correspondiente, luego, pulsando el botón “SHOW ACTION” se desplegará una ventana con la acción escrita en el cuadro de texto (véase figura 3.22). De esta manera podemos pedirle al paciente que la lea y la realice.

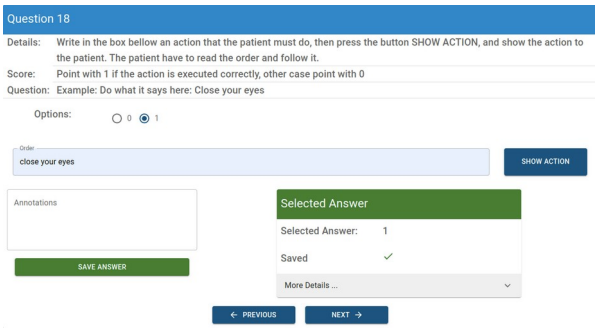

Figura 3.21: Ejemplo de pregunta de tipo realizar acción

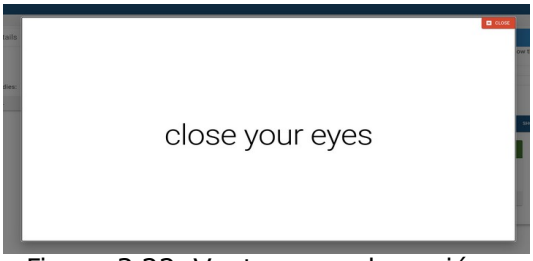

Figura 3.22: Ventana con la acción a realizar por el paciente

- Preguntas de dibujo: Este tipo de pregunta comparte el mismo concepto de la anterior. Pero, en este caso, pulsando el botón “SHOW IMAGE” (véase figura 3.23) , se desplegará una ventana con la imagen ampliada (véase figura 3.24), la cuál tendrá que ser dibujada por el paciente en lápiz y papel. La forma de puntuar el dibujo viene especificada en la propia pregunta. Una vez el paciente haya realizado el dibujo, el evaluador podrá tomar una foto y subirla a la aplicación pulsando el botón “ADD IMAGE” para guardarla en la base de datos como respuesta del paciente.

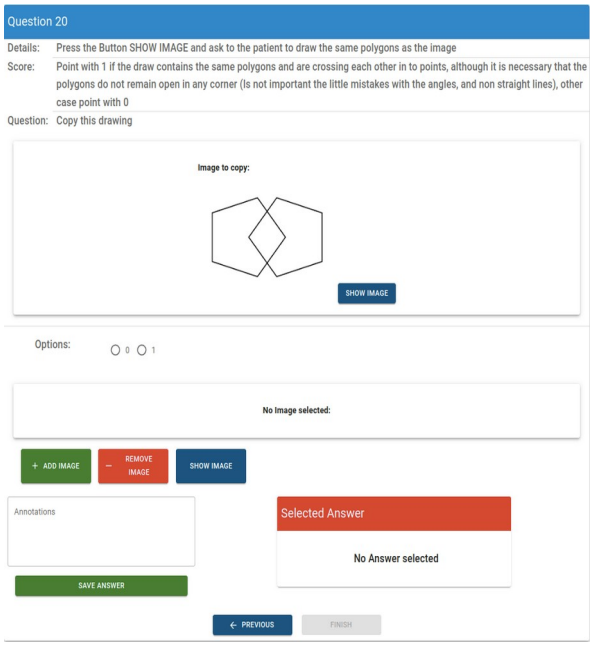

Figura 3.23: Pregunta de dibujo

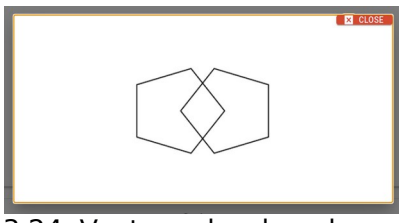

Figura 3.24: Ventana desplegada con la imagen

Una vez terminado el test, la aplicación mostrará la puntuación final del test y dará la posibilidad al usuario de elegir si guardar ese resultado o no (véase figuras 3.25 y 3.26).

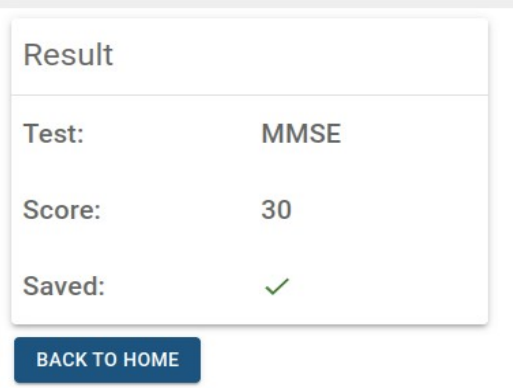

Figura 3.25: Resultado del test guardado

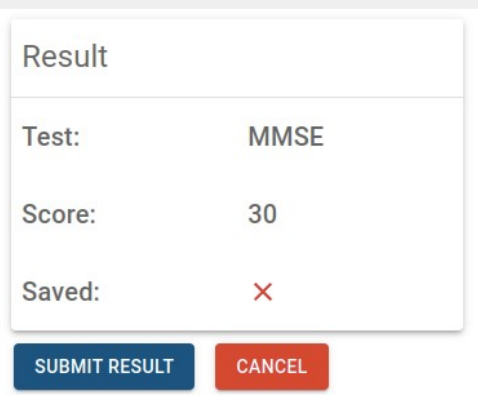

Figura 3.26: Resultado del test sin guardar

3.3.7 Evaluador

Esta vista nos permite realizar las evaluaciones o diagnósticos automáticos de los pacientes en base al conjunto de test y modelos de diagnosticador que se elijan.

Como en todas las vistas, el primer paso es la selección de pacientes, tal como se muestra en la figura 3.27.

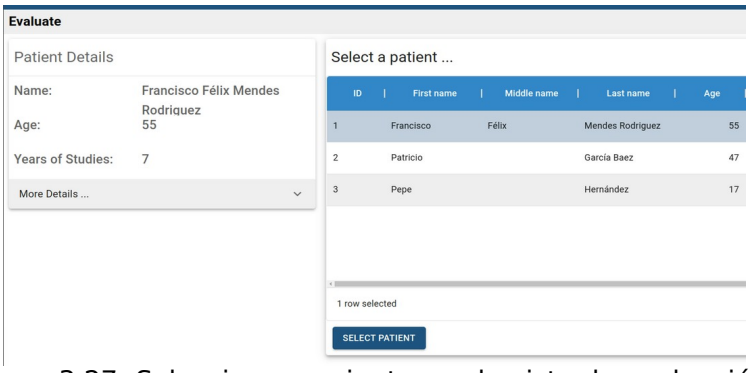

Figura 3.27: Seleccionar pacientes en la vista de evaluación

Una vez seleccionado el paciente accedemos al evaluador (véase figura 3.28). Aquí encontraremos diferentes componentes: una tarjeta con los datos del paciente seleccionado, el botón para volver atrás y seleccionar otro paciente, un selector desplegable para seleccionar los test con los que queremos realizar un diagnóstico con el modelo de inteligencia artificial, una tarjeta donde aparecerán los datos del modelo que realizará la evaluación seleccionado en base a los resultados de los test seleccionados, una tarjeta con los modelos recomendados y ,por último, una tarjeta con los datos de la evaluación realizada.

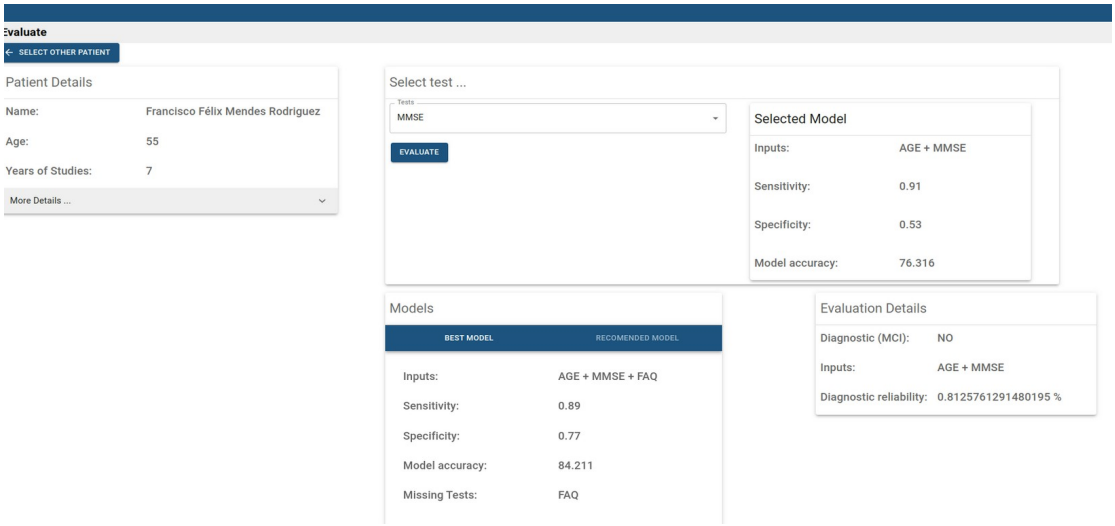

Figura 3.28: Vista del evaluador automático

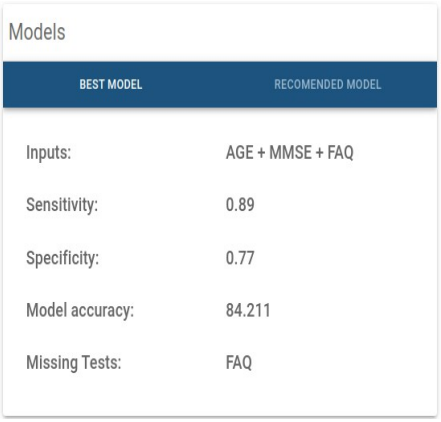

Figura 3.29: Mejor modelo

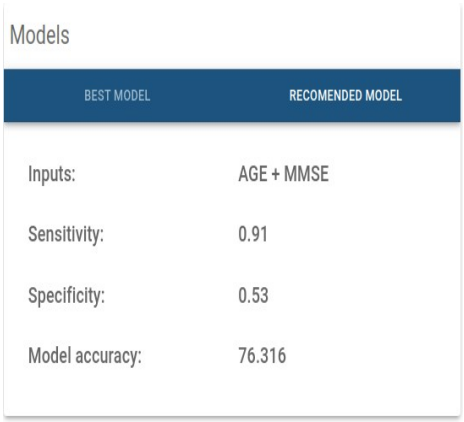

Figura 3.30: Modelo recomendado

En la tarjeta de recomendación tenemos dos pestañas una con el mejor modelo para realizar diagnósticos (figura 3.29), nótese que tenemos un apartado que nos dice que test necesitamos realizar para poder utilizar dicho modelo, en la otra pestaña encontramos el modelo recomendado (figura 3.30) y que es seleccionado por la aplicación en base a los test que tiene realizado el paciente.

3.3.8 Estadísticas

Mediante esta vista podemos acceder a las estadísticas relativas a los test y evaluaciones efectuadas para cada paciente.

Primero seleccionamos el paciente (figura 3.31), como en las vistas de evaluación y de los test que hemos visto previamente.

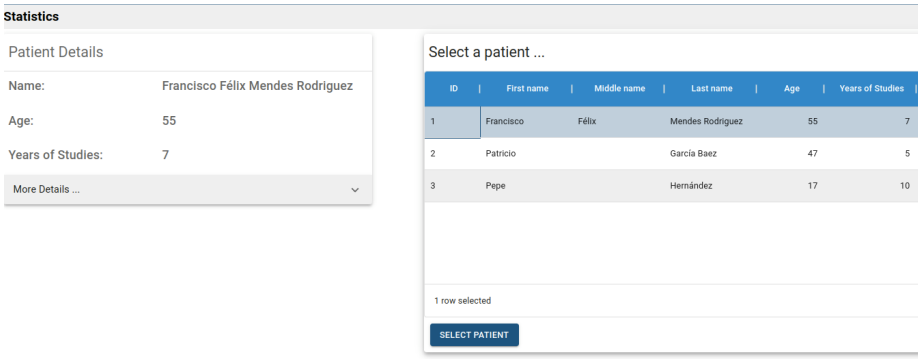

Figura 3.31: Selección de paciente en la vista de estadísticas

Una vez seleccionado el paciente encontramos una suite para la estadísticas (figura 3.32) que cuenta con un botón para volver a seleccionar otro paciente y 3 pestañas:

- **Visualizador de respuestas**: en esta vista podemos seleccionar un test de entre todos los test realizados y poder visualizar las respuestas dadas por el paciente y las anotaciones realizadas por el médico en cada pregunta. Esta pestaña sirve para analizar ese test en concreto más en detalle (véase figura 3.32 y 3.33).

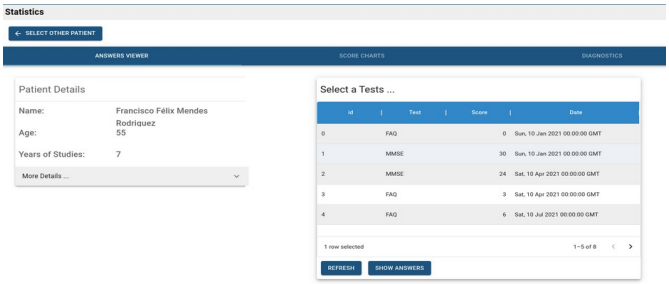

Figura 3.32: Seleccionador de test de la vista de estadísticas

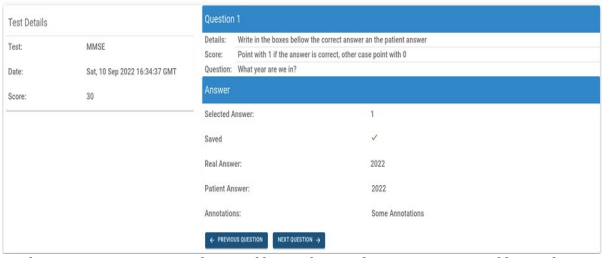

Figura 3.33: Visualizador de test realizados

- **Estadísticas de resultados de los test**: En esta pestaña encontramos dos botones y los detalles del paciente seleccionado. Pulsando el primer botón podremos añadir en la interfaz una nueva gráfica, donde en cada una de ellas podemos seleccionar un test y visualizar un gráfico longitudinal del los resultados del paciente (figura 3.35) para ese test. Se pueden crear tantas gráficas como queramos (limitado al número de test implementados en la aplicación) para que podemos tener diferentes gráficas con los resultados de los diferentes test para hacer comparaciones. Por último tenemos el botón para borrar la gráfica, el cual eliminará la última gráfica que se creó.

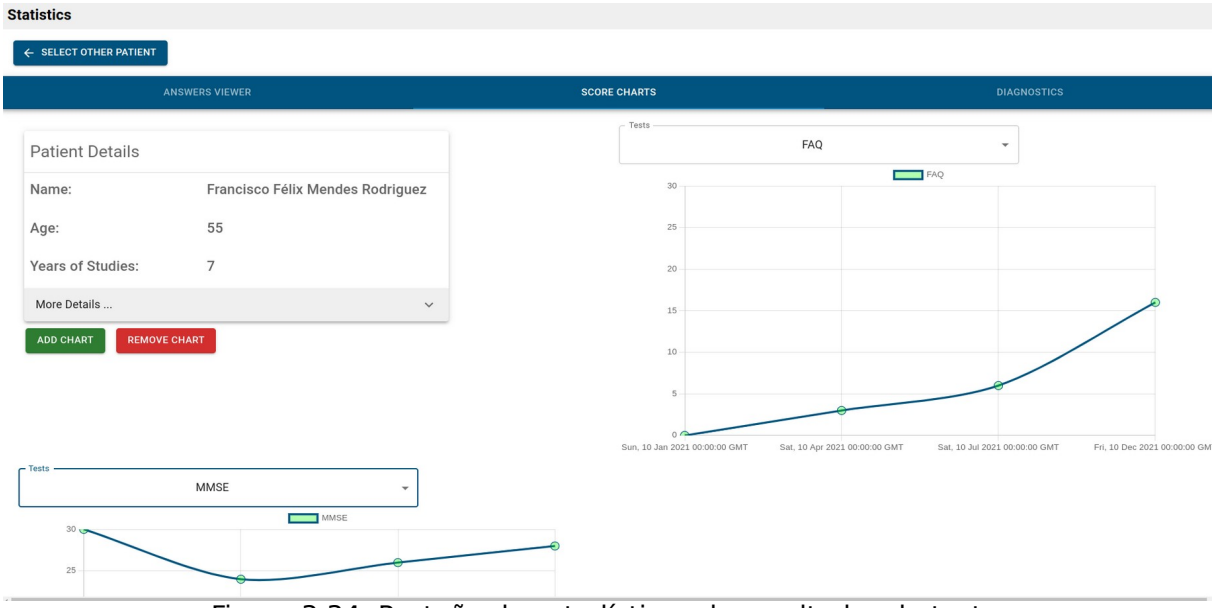

Figura 3.34: Pestaña de estadísticas de resultados de test

- **Estadísticas de diagnósticos automáticos realizados**: Esta pestaña es igual que la anterior con la diferencia que en las gráficas podemos observar el estudio longitudinal de los diagnósticos automáticos realizados al paciente, donde podemos elegir de entre los distintos modelos entrenados el que queremos visualizar, teniendo la posibilidad de tener diferentes gráficos para diferentes modelos y poder compararlos (véase figura 3.35).

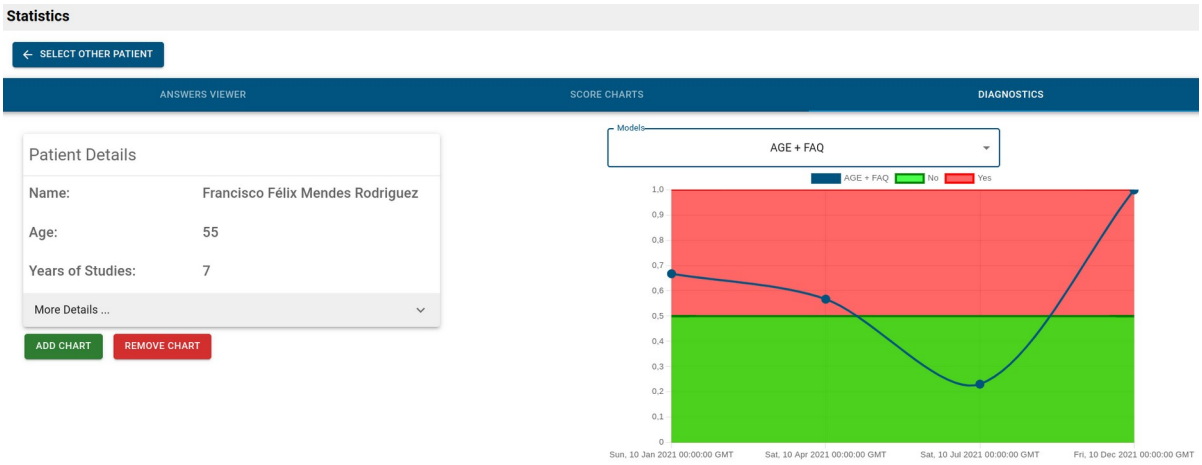

Figura 3.35: Pestaña de estadísticas de diagnósticos

Capítulo 4
Conclusiones

Podemos concluir que este TFG ha sido exitoso ya que se han alcanzado todos los objetivos planteados desde el inicio del proyecto, teniendo como resultado la aplicación NeuroAutoTest, que como ya hemos dicho antes es una plataforma para la realización de test neurocognitivos y el diagnóstico automático de deterioro cognitivo leve. Con esta aplicación se ha brindado a los especialistas en el ámbito de las neurociencias una herramienta, que no los sustituye, sino que sirve como apoyo a la hora de realizar diagnósticos, permitiendo realizar los test neurocognitivos y diagnósticos automáticos de una manera ágil y cómoda. La aplicación es posible ejecutarla sin disponer de un ordenador, ya que puede ser utilizada desde distintos dispositivos como tabletas o móviles.

En lo personal, este proyecto me ha dado la posibilidad de aprender una gran variedad de tecnologías, las cuales me serán útiles en mi vida profesional. También, he aprendidos conceptos nuevos y he afianzado otros aprendidos a lo largo de mi vida universitaria, que me ha hecho sentir que he crecido un poco como profesional. Este proyecto, también me ha brindado la posibilidad de aprender sobre las enfermedades neurodegenerativas, entre otros conceptos relacionados con las neurociencias, que ha despertado en mí un interés en poder contribuir en estos campos en un futuro.

Uno de los aspectos más difíciles de este proyecto fue el aprendizaje de todas las herramientas utilizadas, ya que nunca había trabajado con ellas. También ha sido complicado lidiar con el entrenamiento de los modelos de inteligencia artificial en los que está basado el diagnosticador y el diseño de toda la infraestructura e interfaz de usuario de la aplicación, siempre pensando en tener la mejor experiencia de usuario posible.

Sin duda, la aplicación tiene un gran margen de mejora, llevándonos a plantear las siguientes líneas futuras :

- Mejora de los modelos entrenados, utilizando otros modelos más potentes y/o entrenar los modelos con un conjunto de datos más grande
- Mejora en el diseño de la aplicación, añadiendo funcionalidades como la agrupación de pacientes, un mejor manejo de las estadísticas que brinda la aplicación y un protocolo para realizar los diagnósticos donde el médico tenga más control sobre éste
- Ampliación del catálogo de test que se pueden realizar en la aplicación

- Generación de un modelo de negocio y despliegue de la aplicación, ya que para este TFG se ha decidido desplegar la aplicación de manera local mediante el uso de contenedores Docker, pero sería interesante definir una estrategia para realizar el despliegue de la aplicación de cara a convertirla en un producto real dentro del mercado
- Conexión de la aplicación con otros servicios para la obtención de las historias clínicas de los pacientes, y conseguir así una mejora en la gestión de los pacientes que sea más detallada y brinde la posibilidad de llevar un mejor control sobre los tratamientos y un mejor seguimiento de los pacientes

Capítulo 5
Conclusions

We can conclude that this project has been successful since all the objectives set from the beginning of the project have been achieved, resulting in the NeuroAutoTest application, which, as we have said before, is a platform for performing neurocognitive tests and the automatic diagnosis of mild cognitive impairment. With this application, specialists in the field of neuroscience have been provided with a tool that does not replace them, but rather serves as support when making diagnoses, allowing neurocognitive tests and automatic diagnoses to be carried out in an agile and comfortable way. The application can be run without having a computer, since it can be used from different devices such as tablets or mobile phones.

Personally, this project has given me the opportunity to learn a wide variety of technologies, which will be useful in my professional life. Also, I learned new concepts and I secure others, which has made me feel that I grew as a professional. This project has also given me the opportunity to learn about neurodegenerative diseases, among other concepts related to neuroscience, which has aroused in me an interest in being able to contribute to these fields in the future.

One of the most difficult aspects of this project was learning all the tools used, since I had never worked with them before. It has also been difficult to deal with the training of the artificial intelligence models on which it is based in the diagnosis and design of the entire infrastructure and user interface of the application, always thinking of having the best possible user experience.

Without a doubt, the application has a great margin for improvement, leading us to consider the following future lines:

- Improvement of the trained models, using other more powerful models and/or training the models with a larger data set
- Improvement in the design of the application, adding functionalities such as patient grouping, better management of the statistics provided by the application and a protocol for making diagnoses where the doctor has more control over it.
- Expansion of the test catalog that can be carried out in the application
- Generation of a business model and use of the application, since for this

project it has been decided to deploy the application locally through the use of Docker containers, but it would be interesting to define a strategy to use the application for turn it into a real product in the market

• Connection of the application with other services to obtain the medical records of the patients, and thus achieve an improvement in the management of the patients that is more detailed and offers the possibility of having a better control over the treatments and a better follow-up. from the patients

Capítulo 6
Presupuesto

En esta capítulo hablaremos del presupuesto de la aplicación NeuroAutoTest, en esta ocasión cabe a destacar que la mayoría de las herramientas usadas son de código abierto, aunque las herramientas como Auth0 y Material-UI tienen planes de pagos, pero para el desarrollo de esta aplicación se ha utilizado los planes gratuitos. En cuanto a la mano de obra, estableceremos un precio fijo de 20€ la hora (véase tabla 6.1), ya que es un precio estándar en el salario de un desarrollador Fullstack junior.

6.1 Coste de desarrollo

Concepto	Coste	Horas	Coste total
Desarrollo de la aplicación	20 €/hora	300	6000€

Tabla 6.1: Costes del desarrollo de la aplicación

6.2 Coste de Mantenimiento

En la tabla 6.2 podemos observar los coste de mantenimiento, donde, asumimos que utilizamos Material-UI con su plan Premium, y Auth0 con el plan B2C Professional teniendo una cantidad de usuarios menor de 10000 en nuestra aplicación, y suponiendo que la aplicación necesite desplegarse en 3 servidores (Nginx, Backend, Base de Datos), desplegados en servicios de “hosting”, además de, un personal capacitado de dos personas a jornada completa y un salario mensual de 4500€ para labores de mantenimiento de la aplicación, obtenemos los siguientes costes de mantenimiento:

Concepto	Coste por mes
Material-UI Plan Premium	37€/mes
Auth0 Plan B2C Professional para 10000 usuarios	1500€/mes
Coste mensual del “hosting” de los servidores de (Nginx, Backend, Base de datos)	930€/mes
Coste mensual del personal de mantenimiento	9000€/mes
Total:	11467 €/mes

Tabla 6.2: Costes de mantenimiento de la aplicación

Bibliografía

- [1] N. Gramunt, “¿Qué son las enfermedades neurodegenerativas?”, *El blog de la Fundación Pasqual Maragall, 29-06-*2021. [En Línea]. Disponible en: https://blog.fpmaragall.org/enfermedades-neurodegenerativas. (Accedido: 07-sep-2022).
- [2] G. Sabina Sánchez, “Detección del deterioro cognitivo leve. Una propuesta basada en redes neuronales artificiales híbridas,” Trabajo de Fin de Grado, Univ. Las Palmas de Gran Canarias, Gran Canarias, España, 2021. [En línea]. Disponible en: http://hdl.handle.net/10553/103595.
- [3] A. García Rodríguez, “Cogmultest: Aplicación web y móvil para la gestión de múltiples test neurocognitivos,” Trabajo de Fin de Grado, Univ. La Laguna, Tenerife, España, 2015. [En línea]. Disponible en: https://riull.ull.es/xmlui/handle/915/849?show=full.
- [4] GmbH , “Sitio web de Neotiv”. [En línea]. Disponible en: https://neotiv.com/en
- [5] J.Y.C. Chan, S.T.Y. Yau, T.C.Y. Kwok y K.K.F. Tsoi, " Diagnostic performance of digital cognitive tests for the identification of MCI and dementia: A systematic review," Ageing Research Reviews, vol.72, no. 101506, dic. 2021. [En línea]. Disponible en: https://www.sciencedirect.com/science/article/pii/S1568163721002531.
- [6] I. Alexandre Soriano, “Diseño e implementación de un clasificador para la detección de la enfermedad de Alzheimer mediante el uso técnicas de Inteligencia Artificial en imágenes de resonancia magnética”, Trabajo de Fin de Grado, Univ. Politécnica de Valencia, Valencia, España, 2020. [En línea]. Disponible en: http://hdl.handle.net/10251/156295.
- [7] MedlinePlus en español, Bethesda (MD): Biblioteca Nacional de Medicina (EE. UU.) , “Pruebas cognitivas”, 10-12-2020. [En Línea]. Disponible en: https://medlineplus.gov/spanish/pruebas-de-laboratorio/pruebas-cognitivas/.(Accedido: 07-sep-2022).
- [8] S.A. Greenberg, “The Geriatric Depression Scale (GDS)”, 2012. [En línea]. Disponible en: https://wwwoundcare.ca/Uploads/ContentDocuments/Geriatric%20Depression%20Scale.pdf. (Accedido: 07-sep-2022).
- [9] A.M. Mayo, “Use of the Functional Activities Questionnaire in Older Adults with Dementia”, 2016. [En línea]. Disponible en: https://wwwoundcare.ca/Uploads/ContentDocuments/Geriatric%20Depression%20Scale.pdf. (Accedido: 07-sep-2022).
- [10] D. Feijoo, E. Ginesta, A.M Alambiaga-Caravaca, M Azorín, E. Córcoles, J. Botella, M. Alacreu, M.T. Climent,L. Moreno. [En línea]. Disponible en:https://www.farmaceuticoscomunitarios.org/anexos/vol11_n1/ANEXO2.pdf. (Accedido: 07-sep-2022).
- [11] DataScientest, “Perceptrón: ¿qué es y para qué sirve?”, *DataScientest, 7-03-2022. [En línea]. Disponible en:* https://datascientest.com/es/perceptron-que-es-y-para-que-

sirve#:~:text=Un%20perceptr%C3%B3n%20es%20una%20neurona%20artificial%2C%20y%2C%20por%20tanto%2C,aprendizaje%20supervisado%20de%20clasificadores%20binarios. (Accedido: 07-sep-2022).

- [12] C. Bento, “Multilayer Perceptron Explained with a Real-Life Example and Python Code: Sentiment Analysis”, 21-09-2021. Disponible en: https://towardsdatascience.com/multilayer-perceptron-explained-with-a-real-life-example-and-python-code-sentiment-analysis-cb408ee93141. (Accedido: 07-sep-2022).

- [13] Wikipedia contributors, “Backpropagation”, *Wikipedia, The Free Encyclopedia.* 17-08-2022. [En línea]. Disponible en: https://en.wikipedia.org/w/index.php?title=Backpropagation&oldid=1104872812. (Accedido: 08-sep-2022).

- [14] Khan Academy, “Descenso del gradiente”, *Khan Academy*, [En línea]. Disponible en: https://es.khanacademy.org/math/multivariable-calculus/applications-of-multivariable-derivatives/optimizing-multivariable-functions/a/what-is-gradient-descent. (Accedido: 07-sep-2022).

- [15] TIBCO, “¿Qué es el aprendizaje supervisado?”, *TIBCO.* [En línea]. Disponible en: https://www.tibco.com/es/reference-center/what-is-supervised-learning#:~:text=El%20aprendizaje%20supervisado%20es%20una,de%20manera%20expl%C3%ADcita%20d%C3%B3nde%20buscar. (Accedido: 08-sep-2022).

- [16] MDN contributors, “JavaScript” , *MDN.* 04-08-2022. [En línea]. Disponible en: https://developer.mozilla.org/es/docs/Web/JavaScript. (Accedido: 08-sep-2022).

- [17] A. Visus, “¿Para qué sirve Python? Razones para utilizar este lenguaje de programación”, *ESIC,* oct. 2020. [En línea]. Disponible en: https://www.esic.edu/rethink/tecnologia/para-que-sirve-python. (Accedido: 08-sep-2022).

- [18] Facebook Open Source, “Página oficial de React”, *React*, [En línea]. Disponible en: https://es.reactjs.org/. (Accedido: 08-sep-2022).

- [19] Okta. Inc, “Portal oficial de Auth0”, *Auth0.* [En línea]. Disponible en: https://auth0.com/es?utm_source=google&utm_medium=cpc&utm_term=-g-auth0&pm=true&utm_campaign=spain-es-brand-auth0&gclid=Cj0KCQjwof6WBhD4ARIsAOi65ajXF-rcz3E1Fmx2VHTWSl2rZGFv4FonwhrSAtQPxcXIxrZQKSkciUkaAmd_EALw_wcB. (Accedido: 08-sep-2022).

- [20] J. Dominguez Muñoz, “¿Qué es Flask?”, *OpenWebinars*. 17-11-2017. [En línea]. Disponible en: https://openwebinars.net/blog/que-es-flask/. (Accedido: 08-sep-2022).

- [21] RedHat. Inc, “¿Qué es una API de REST?”, *RedHat*, 08-05-2020. [En línea]. Disponible en: https://www.redhat.com/es/topics/api/what-is-a-rest-api. (Accedido: 08-sep-2022).

- [22] Wikipedia Contributors, “Protocolo de transferencia de hipertexto”, *Wikipedia, The Free Encyclopedia.* 08-10-2022. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=Protocolo_de_transferencia_de_hipertexto&oldid=145840173. (Accedido: 09-sep-2022).

- [23] G. Bustos. “¿Qué es un CDN? Content Delivery Network”, *Hostinger*. 24-08-2022. [En línea]. Disponible en: https://www.hostinger.es/tutoriales/que-es-cdn?ppc_campaign=google_performance_max&gclid=Cj0KCQjwpeaYBhDXARIsAEzItbEp_MS5W9s8xI3Mi_twDyKgjCTTAyEJNV_nXryaguHIY84ZTihogjAaAmh7EALw_wcB. (Accedido: 09-sep-2022).

- [24] P. Araneda. *Base de Datos.* Bookdown, 20-12-2020. [En línea]. Disponible en: https://bookdown.org/paranedagarcia/database/el-modelo-relacional.html.

- [25] The PostgreSQL Global Development Group, “Página oficial de PostgreSQL”. [En línea]. Disponible en: https://www.postgresql.org/
- [26] Wikipedia Contributors, “SQL”, *Wikipedia, The Free Encyclopedia.* 27-10-2022. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=SQL&oldid=145610436. (Accedido: 09-sep-2022).
- [27] Google Colab, “Colaboratory”. [En línea]. Disponible en: https://research.google.com/colaboratory/intl/es/faq.html
- [28] G. Bello, “Introducción al cuaderno Jupyter para principiantes”, *Geekflare*. 12-11-2019. [En línea]. Disponible en: *https://geekflare.com/es/jupyter-notebook-basics/.* (Accedido: 10-sep-2022).
- [29] Scikit-Learn, “Página oficial de Scikit-learn”. [En línea]. Disponible en: https://scikit-learn.org/stable/
- [30] D. Carolina, “¿Qué es Docker y cómo funciona? – Una explicación sencilla”, *Hostinger,* 30-06-2022. [En línea]. Disponible en: https://www.hostinger.es/tutoriales/que-es-docker?ppc_campaign=google_performance_max&gclid=Cj0KCQjwpeaYBhDXARIsAEzItbGmXXKZR_sx-UwE0idv31d2II59ALT-DLlRC9JgiRzS-eidfL-cUFlaArjREALw_wcB. (Accedido: 10-sep-2022).
- [31] Psycog. “Página oficial de Psycog”. [En línea]. Disponible en: https://www.psycopg.org/docs/
- [32] OpenJS Foundation and Node.js contributors. “Página oficial de Node.js”. [En línea]. Disponible en: https://nodejs.org/es/about/
- [33] Axios, “Página oficial de Axios”. [En línea]. Disponible en: https://axios-http.com/es/docs/intro
- [34] Material UI, “Página oficial de Material-UI”. [En línea]. Disponible en: https://mui.com/
- [35] “Página oficial de JSON”. [En línea]. Disponible en: https://www.json.org/json-es.html
- [36] AbrirArchivos, “.pkl Extensión de archivo”. [En línea]. Disponible en: https://abrirarchivos.info/extension/pkl
- [37] Alzheimer’s Disease Neuroimaging Initiative. “Sitio web de ADNI”. [En línea]. Disponible en: https://adni.loni.usc.edu/
- [38] C.P. Suárez-Araujo, P. García Báez, Y. Cabrera-León, A. Prochazka, N. Rodríguez Espinosa, C. Fernández Viadero, for the Alzheimer’s Disease Neuroimaging Initiative, "A Real-Time Clinical Decision Support System, for Mild Cognitive Impairment Detection, Based on a Hybrid Neural Architecture”, Computational and Mathematical Methods in Medicine, vol. 2021, Article ID 5545297, 2021. [En línea]. Disponible en: https://doi.org/10.1155/2021/5545297
- [39] R. Cañadas, ”Curvas ROC”, *AbDatum.* 28-08-2021. [En línea]. Disponible en: https://abdatum.com/ciencia/curvas-roc