UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA BIOMÉDICA

TRABAJO FIN DE GRADO

DISEÑO E IMPLEMENTACIÓN DE APLICACIONES WEB PARA LA DIVULGACIÓN Y VISUALIZACIÓN DE HERRAMIENTAS DE EVALUACIÓN DE PACIENTES CON DÉFICIT MOTOR

ADRIÁN DE SOUSA DA SILVA

2023

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA BIOMÉDICA

TRABAJO FIN DE GRADO

DISEÑO E IMPLEMENTACIÓN DE APLICACIONES WEB PARA LA DIVULGACIÓN Y VISUALIZACIÓN DE HERRAMIENTAS DE EVALUACIÓN DE PACIENTES CON DÉFICIT MOTOR

Autor

ADRIÁN DE SOUSA DA SILVA

Tutores

ÁLVARO GUTIÉRREZ MARTÍN
MARÍA ALONSO DE LECIÑANA

2023

Resumen

Actualmente existen diferentes enfermedades neurológicas que pueden causar una serie de secuelas en los pacientes que las padecen. Una de las secuelas de estas enfermedades es el déficit motor, una de las causas más comunes de discapacidad que puede cambiar por completo la vida de una persona. Por ello, es necesario abordar esta problemática con herramientas innovadoras que ayuden a los profesionales de la salud a obtener mejores resultados en cuanto a tratamientos y la recuperación del paciente.

En este Trabajo Fin de Grado se ha desarrollado una aplicación web dinámica para la visualización de datos de pacientes con déficit motor, como un aporte a las herramientas mencionadas anteriormente. Dicha aplicación se ha diseñado con el objetivo de cuantificar el déficit motor y permitir mejoras significativas en diagnósticos, monitorizaciones, seguimientos y tratamientos en el área de la neurología, en la que se podrá contar con una herramienta nueva que posee una interfaz de usuario intuitiva y fácilmente manejable.

Para cumplir con el objetivo, la aplicación es capaz de conectarse a una base de datos, alojada en el servidor del Laboratorio de Robótica y Control de la ETSIT-UPM, que contiene los datos de cada paciente que ha realizado una serie de ejercicios para permitir la cuantificación del déficit y la generación de los datos usados por esta herramienta. Además, cuenta con un *backend* para la recuperación y procesado de dichos datos. También posee un *frontend* que presenta todos elementos gráficos de la aplicación mediante el cual los profesionales de la salud podrán acceder a los datos de los pacientes a través de la red.

Además, se ha creado un módulo de generación de contenido web automático con el objetivo de divulgar información sobre una serie de proyectos de innovación tecnológica relacionados con enfermedades neurológicas. Estos se han desarrollado bajo la colaboración de las instituciones del Hospital Universitario La Paz, Instituto de Investigación IdiPAZ, la Escuela Técnica Superior de Ingenieros de Telecomunicación y el Laboratorio de Robótica y Control de la ETSIT, en busca de ampliar el público general al que se dirigen estas tecnologías.

Palabras clave: Aplicación web, *frontend*, *backend*, datos, pacientes.

Abstract

Nowadays there are several different neurological diseases that may cause sequels in patients who suffer from this kind of diseases. Motor deficit is among those sequels and it is one of the most common causes of disability, which can change people's lives entirely. It is therefore that this problematic situation needs to be addressed with new innovative technological tools that help the healthcare professionals to get better results from treatments and rehabilitation.

In this bachelor's thesis has been developed a dynamic web application for motor deficit patients' data visualization, as one of the innovative and technological tools previously mentioned. This application has been designed for motor deficit quantification, and that creates new ways of improving diagnoses, monitoring and treatments in neurology's department, which will have the opportunity to count with a new tool that has a very intuitive and manageable user interface.

To accomplish the aim of this project, the application establish a connection with a database, stored in Laboratorio de Robótica y Control's web server, which contains the information of all the patients who did several exercises to allow the deficit quantification and generated the data used by this tool. Also, the application has a backend that search for the information and process it. It also has a frontend that shows all graphic elements used by healthcare professionals to access to patients' data via network.

Besides, it has been created an automatic module for web content with the aim of publicize information about several innovative technological projects related to neurological diseases. Hospital Universitario La Paz, Instituto de Investigación IdiPAZ, ETSIT-UPM and Laboratorio de Robótica y Control from ETSIT are the collaborative institutions which make possible to develop those projects, in order to broaden the audience to which these technologies are addressed.

Keywords: Web application, frontend, backend, data, patients

Agradecimientos

En primer lugar, deseo mostrar mi agradecimiento al Hospital Universitario La Paz, al Instituto de Investigación Hospital Universitario La Paz, al Laboratorio de Robótica y Control y a la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universidad Politécnica de Madrid por hacer posible el desarrollo de este Trabajo Fin de Grado bajo la colaboración de dichas instituciones. Además de mis tutores, Álvaro Gutiérrez Martín y María Alonso de Leciñana.

También quiero agradecer a mi familia, sobretodo a mis padres, Gabriela Da Silva y Pascual De Sousa, a mi hermano, Gabriel De Sousa y a mi novia Anna Ocampo por haber sido un gran apoyo durante el grado y por creer en mi en todo momento, incluso cuando yo no lo hacía. Y a mis abuelos por ser una motivación en mi vida y por sus enseñanzas.

De igual manera a mi grupo de amigos del colegio que me han acompañado y apoyado durante todos estos años de esfuerzo y aprendizaje. Además, a mis amigos de la universidad por estos cuatro años inolvidables en los que nos hemos apoyado el uno al otro en todo momento y hemos formado amistades que son para toda la vida.

Y por último, agradezco mis habilidades y competencias que me han llevado hasta aquí.

Índice general

Índice de figuras

Índice de tablas

Lista de Acrónimos

UPM: Universidad Politécnica de Madrid.

ETSIT: Escuela Técnica de Ingenieros de Telecomunicación.

ROBOLABO: Laboratorio de Robótica y Control.

HULP: Hospital Universitario La Paz.

HTML: HyperText Markup Language.

CSS: Cascading Style Sheets.

JS: JavaScript.

JSON: JavaScript Object Notation.

MVC: Modelo Vista Controlador.

CMS: Content Manager System.

ECV: Enfermedad cerebrovascular.

API: Application Programming Interface

Capítulo 1

Introducción

Este Trabajo Fin de Grado surge como un proyecto en colaboración entre el Laboratorio de Robótica y Control de la ETSI Telecomunicación y el Instituto para la Investigación Biomédica del Hospital Universitario La Paz-IdiPAZ, y tiene el objetivo de implementar una aplicación web para la visualización de datos de pacientes con déficit motor. El TFG se centra en desarrollar una herramienta que permita a los profesionales de la salud realizar evaluaciones, diagnósticos y seguimientos más precisos. Además, esta aplicación web es una forma sencilla de acceder y representar los datos, medidos y calculados, de los pacientes a través de una interfaz de usuario sencilla e intuitiva.

También se abarca un área de divulgación de información sobre diferentes proyectos tecnológicos innovadores relacionados con algunas problemáticas del campo de la neurología, con la finalidad de lograr una mayor diseminación al público general.

1.1. Aplicaciones web

Las aplicaciones web son programas que ejercen sus funciones en Internet. No requieren de instalación en los dispositivos y los recursos que utilizan son procesados en la web. Luján (2002) define a una aplicación web como "un tipo especial de aplicación cliente/servidor, donde tanto el cliente como el servidor y el protocolo mediante el que se comunican están estandarizados y no han de ser creados por el programador de aplicaciones" [1].

Hoy en día cumplen un papel muy importante dentro de cualquier ámbito ya que son herramientas que ayudan a los profesionales a obtener mejores resultados con mayor eficiencia.

1.1.1. Tipos de aplicaciones web

El desarrollo de este TFG se centrará en el uso de tres tipos diferentes de aplicaciones web: aplicaciones web estáticas, dinámicas y de portal.

En la actualidad existe una gran variedad de aplicaciones web que varían según la información que muestran, lenguajes de programación utilizados para su construcción y el objetivo que tienen respecto a los usuarios que las utilizan. Se pueden clasificar en [2]:

- **Aplicaciones web estáticas**: las aplicaciones web estáticas son el tipo más simple ya que solo están destinadas a mostrar una información especifica, sin la posibilidad de variar su contenido, efectos especiales o interacciónes con el usuario. Los más simples se componen a de ficheros HTML y CSS.
- **Aplicaciones web dinámicas**: las aplicaciones web dinámicas tienen como característica principal la variabilidad de su contenido. Además, permite a los usuarios interactuar con ella y actualizar datos sin la necesidad de editar la información [3]. Para la construcción de estas aplicaciones se suelen usar lenguajes básicos de desarrollo web como HTML y CSS, pero con la adición de otros lenguajes como PHP, JavaScript, Asp u otros.
- **Aplicaciones web de comercio electrónico**: son aplicaciones web dinámicas destinadas a la comercialización de productos y/o servicios de empresas o personas que las utilizan. Deben ser capaces de mostrar la información de lo que se ofrece, administrar pagos, generar facturas electrónicas e implementar métodos de contacto como correo electrónico, formularios o mensajes en vivo.
- **Aplicaciones web de gestión de contenido**: también conocidas por sus siglas en inglés como CMS (Content Manager System), son aplicaciones web que funcionan como intermediarios entre el desarrollador web y su aplicación. Cumplen con la administración de contenidos que se muestran en un sitio web.
- **Aplicaciones web de página única**: son aplicaciones web que se basan en la constante actualización de una página y que todas sus acciones se realizan en la misma ventana del navegador.
- **Aplicaciones web de portal**: las aplicaciones web de portal son aquellas destinadas a redirigir a los usuarios a otras páginas web. Mediante enlaces se redirecciona a los visitantes a otros servicios o sitios web, que se suelen abrir en otra ventana. Además, pueden ser securizadas mediante un sistema de usuarios o de suscripciones.
- **Aplicaciones web progresivas**: son aplicaciones web que tienen el objetivo de ofrecer una experiencia igual o parecida a la de una aplicación que solo está disponible en tiendas oficiales. Integran la experiencia de navegación y el uso de interfaces que se acercan a las de uso móvil.

1.2. Generación y gestión de datos en la industria

La última década del siglo XX fue una época de cambios para la comunicación científica. El aumento de los precios de revistas científicas, restricciones legislativas sobre derechos de autor y la diseminación de información y demás problemas relativos al sistema tradicional de comunicación científica, desencadenó una crisis en dicho

sistema que desfavorecía la diseminación y el intercambio de resultados científicos. En respuesta a esta problemática, nació el movimiento *Open Access* (OA) [4].

En 2013, Torrecilla describió dicho movimiento: Promueve la libre disponibilidad pública en Internet de los documentos de investigación científica, permitiendo a cualquier usuario la lectura, descarga, copia, distribución, impresión, búsqueda, o el vínculo a los textos completos de dichos artículos, la única restricción es dar a los autores control sobre la integridad de su trabajo y el derecho a ser reconocidos y citados ([4], p.7).

Dicho movimiento fue discutido y editado en años posteriores y es el responsable de que Internet y sus medios de comunicación asociados como páginas web o blogs cumplan un papel muy importante en la comunicación de conocimientos y datos científicos, así como su sostenibilidad a lo largo del tiempo. Aedmás, nace el concepto de *datos de investigación* [4].

Los datos de investigación, definidos por la Organización para la Cooperación Económica y el Desarrollo (OECD), son "todos los materiales registrados durante la investigación, reconocidos por la comunidad científica y que sirven para certificar los resultados de la investigación que se realiza". Dichos datos pueden utilizarse para generar nuevos conocimientos o validar resultados de otras investigaciones [4].

Los avances tecnológicos y la problemática descrita anteriormente han generado la necesidad de conservar grandes cantidades de datos de investigación que cada vez son más complejos y que representan la base fundamental de las investigaciones actuales y futuras [4].

La industria moderna se encuentra en una era de digitalización de la información, en la que la recopilación, tratamiento y almacenamiento de los datos es vital para asegurar, en la medida de lo posible, la innovación y el éxito de futuras investigaciones, productos y servicios destinados a diferentes áreas profesionales y sociales.

En la actualidad, los datos son considerados la nueva materia prima del siglo XXI debido a que representan un activo económico para cualquier sector profesional de la actualidad [5].

1.2.1. Gestión de datos en la industria de la salud

Se estima que técnicas de procesamiento de datos basadas en *Big Data* ofrecerán "nuevas posibilidades para entender mejor la biología de sistemas en los humanos así como para desarrollar herramientas de medicina personalizada, previsión de enfermedades y, en definitiva, mejorar la salud de las personas"(Bilbao et. al., s.f.)[6]. La historia o registro es la principal fuente de información de donde se toman los datos de cada paciente, aunque existen otros tipos de recopilación de datos que incluyen avances tecnológicos como aplicaciones móviles o redes sociales [6].

Campos de la salud como el diagnóstico por imagen, pruebas de laboratorio, proteómica, emergencias y estimación de enfermedades han implementado tecnologías de inteligencia artificial relacionadas a la toma de decisiones que se basan en el procesado y análisis de los datos que se obtienen de los pacientes [7]. En el caso particular de este TFG se desarrolla una aplicación web que desempeña sus funciones en el campo de la neurología como una herramienta de evaluación del déficit motor que tiene la capacidad de procesar datos relacionados a dicha temática y representarlos gráficamente.

1.3. Déficit motor y su impacto en pacientes

En el área de la salud existen una gran cantidad de enfermedades neurológicas que se pueden estudiar a partir de tecnologías de sistemas informáticos, técnicas de análisis de datos y la inteligencia artificial. Estos avances tecnológicos se han implementado para dar lugar a nuevos abordajes en la práctica clínica del campo de la neurología [8]. Hay diferentes consecuencias ocasionadas por estas enfermedades, siendo el déficit motor una de ellas.

El déficit motor es una de las causas más comunes de discapacidad en pacientes y una de las más perjudiciales para la calidad de vida, ya que afecta la movilidad de la persona, limita el desarrollo normal de las actividades cotidianas y dificulta la reinserción social y/o profesional [9]. Una de las enfermedades más comunes que causa déficit motor en una gran cantidad de pacientes es el ictus, la cual afecta entre 110.000 a 120.000 pacientes cada año solo en España, de los cuales el 50 % quedan con secuelas discapacitantes o mueren [10].

El ictus, también conocido como enfermedad cerebrovascular (ECV), es un "trastorno circulatorio cerebral que ocasiona una alteración transitoria o definitiva de la función de una o varias partes del encéfalo"(Ustrell-Roig y Serena-Leal, 2007)[11]. Es la primera causa de discapacidad adquirida en la edad adulta y la segunda causa de muerte en países desarrollados. Se clasifica en dos tipos, isquémico y hemorrágico. El primero se debe a una interrupción del aporte sanguíneo de una parte concreta del parénquima encefálico, mientras que el segundo se origina por la rotura de un vaso sanguíneo de encéfalo seguido de la extravasación de la sangre [11].

Más del 80 % de pacientes que han sufrido un ictus padecen hemiparesia como una de las consecuencias que afectan la motricidad, y 40 % de ellos de forma crónica [9]. Por ello es importante tener las herramientas adecuadas para realizar una cuantificación, monitorización y seguimiento precisos y exhaustivos de la evolución del déficit en el paciente, para que la mayoría de ellos puedan recuperar sus cualidades motoras en la mayor medida posible.

1.3.1. Tecnologías para abordar el déficit motor en pacientes

Los profesionales de la salud ejercen una gran labor para detectar problemas de salud en los pacientes, abordarlos de la mejor manera posible y realizar un seguimiento en su evolución hasta la recuperación completa o parcial. Pero en la actualidad se han desarrollado avances tecnológicos que complementan estas tareas para hacerlas más eficaces y eficientes, además de obtener resultados mejores y más precisos.

En la actualidad existen avances tecnológicos basados en aplicaciones, inteligencia artificial, robótica, genómica, telemedicina, entre otros. Estas herramientas son utilizadas en el área de la salud con la finalidad de mejorar diágnositicos, la toma de decisiones, tratamientos y demás cuidados para los pacientes [12].

Existen tecnologías capaces de cuantificar el déficit motor a través de la captación de variables cinemáticas de los pacientes a la hora de realizar diferentes ejercicios que requieren de distintos movimientos del mismo. En el estudio realizado por Gutiérrez et. al (2021) se utiliza esta clase de tecnología en pacientes que han sufrido ictus y se obtuvieron resultados que destacan la utilidad de esta herramienta para detectar niveles muy pequeños de déficit motor pero que aún así afectan la calidad de vida [13]. Esto es una prueba de que la tecnología es vital en el campo de la medicina actual para innovar y mejorar tanto los procesos como los resultados.

Además, el grupo colaborativo formado por la ETSIT-UPM y el IdiPaz poseen una gran variedad de proyectos en el área de investigación y estudio del déficit motor, en los que se desarrollan herramientas que se desempeñan en actividades de I+D+i en el campo de la neurología y que dan la posibilidad de obtener grandes cantidades de información. Dichos proyectos se presentan en el módulo de generación de contenido web.

En este TFG se presenta una aplicación web de visualización de datos cinemáticos de pacientes que padecen déficit motor con la finalidad de ayudar a médicos a realizar seguimientos más precisos de la evolución. Se espera que sea una herramienta que cumpla con lo anteriormente descrito acerca de avances tecnológicos que son utilizados por los profesionales de la salud.

1.4. Objetivos

El propósito de este Trabajo de Fin de Grado es el diseño e implementación de una aplicación web para la visualización de datos de pacientes con déficit motor, una herramienta que ayude a los profesionales de la salud a realizar evaluaciones, diagnósticos y seguimientos más precisos sobre el déficit motor. También se abarca un área de divulgación de información sobre diferentes proyectos tecnológicos innovadores que se exponen mediante aplicaciones web estáticas para lograr una mayor diseminación al público general. Para cumplir con esta finalidad se han establecido diferentes objetivos,

que se han dividido en dos partes, una correspondiente a la visualización de datos y la otra a la divulgación de herramientas. Dichos objetivos se exponen a continuación:

- Aplicación web dinámica para la visualización de datos:
  1. Desarrollo de un back-end que se comunica con una base de datos y con el front-end de la aplicación.
  2. Desarrollo de un front-end que se encarga de mostrar la información de manera apropiada en cada vista.
- Módulo de generación de contenido web automático:
  1. Desarrollo de plantillas web basadas en HTML, CSS y JSON para estructurar y estilizar las aplicaciones web estáticas.
  2. Desarrollo de una plantilla de código JavaScript para automatizar la generación de las páginas web.

1.5. Planificación del proyecto

Este Trabajo de Fin de Grado se ha divido en 5 capítulos, tomando en cuenta los objetivos descritos en la sección anterior. Dichos capítulos de comentan a continuación:

- **Capítulo 1 - Introducción:** capítulo introductorio en el que se expone el concepto de aplicación web y los tipos que existen, se contextualiza la importancia de la gestión de datos en la actualidad y se comenta el impacto del déficit motor en los pacientes y tecnologías que lo abordan.
- **Capítulo 2 - Desarrollo de una aplicación web de visualización de datos:** se describe el desarrollo de la aplicación, las tecnologías empleadas y su funcionamiento. Se comentan aspectos importantes acerca de la base de datos involucrada y la información que contiene. Se presentan los resultados obtenidos.
- **Capítulo 3 - Resultados de la aplicación web de visualización de datos:** se presentan los resultados de la aplicación web de visualización de datos, en los que se incluyen las gráficas que muestra la aplicación, además de los tiempos de carga de dichas gráficas.
- **Capítulo 4 - Módulo de generación de contenido web:** se describe la estructura del módulo, las tecnologías empleadas y el funcionamiento tanto del *frontend* como del *backend*.
- **Capítulo 5 - Conclusiones y líneas futuras:** por último se presentan las conclusiones y líneas futuras.

Capítulo 2

Desarrollo de una aplicación web de visualización de datos

En este capítulo se explica detalladamente el desarrollo y funcionamiento de una aplicación web de visualización de datos basada en la información recopilada por un dispositivo que se menciona en el apartado de resultados de este capítulo.

2.1. Descripción de la aplicación

Es una aplicación web dinámica que tiene la capacidad de procesar datos y de representarlos gráficamente en una interfaz de usuario muy inituitiva de fácil manejo. Está diseñada y desarrollada con el fin de ayudar a los profesionales de la salud del área de neurología a realizar evaluaciones, diagnósticos y seguimientos más precisos sobre aquellos pacientes que padecen déficit motor como secuela de una enfermedad neurológica como el ictus. Además, representa una forma muy sencilla de acceso a los datos de los pacientes involucrados, ya que se puede acceder a la aplicación desde cualquier navegador.

2.1.1. Arquitectura y tecnologías utilizadas

La arquitectura utilizada en la aplicación se basa en el patrón Modelo-Vista-Controlador (MVC)(ver Figura 2.1). Este es un modelo de arquitectura que se caracteriza por tratar al Modelo, la Vista y el Controlador como entidades separadas [14]. Dichos elementos se explican a continuación [15]:

- **Modelo:** contiene las estructuras de datos.
- **Vista:** componentes gráficos que muestran la información al usuario. Se actualiza dependiendo del estado del modelo.
- **Controlador:** controles de la interfaz que alteran el modelo cuando el usuario realiza un evento (clic en un botón, presionar una tecla y otros).

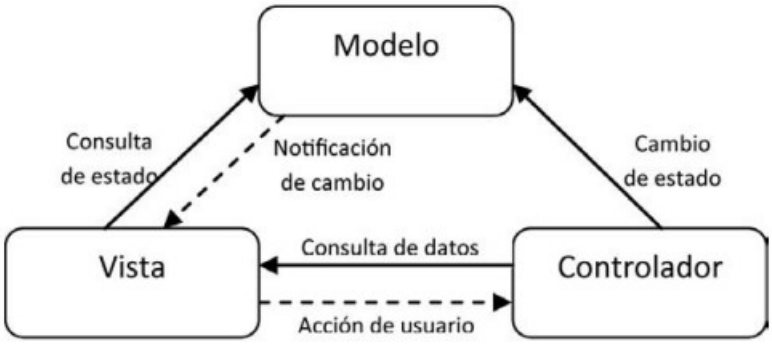


Figura 2.1: Modelo Vista Controlador (tomado de: [15]).

A continuación se explican detalladamente las tecnologías utilizadas, tanto de *frontend* como de *backend* y en qué parte del patrón MVC se desempeñan.

2.1.1.1. Modelo

- **MongoDB:** es un sistema de gestión de bases de datos no relacionales. Proporciona un modelo de almacenamiento de datos elástico que permite almacenar diferentes tipos de datos y consultarlos fácilmente. Una de las grandes ventajas que ofrece es el soporte de distintos lenguajes de programación [16]. Entre dichos lenguajes se encuentra Python, el lenguaje utilizado para el desarrollo del *backend* de la aplicación.
- **MongoDB Compass:** es una interfaz gráfica que sirve para consultar, optimizar y analizar datos en MongoDB [17]. En este caso se ha utilizado como una guía para visualizar la estructura de los documentos guardados en la base de datos.

2.1.1.2. Vista

- **HTML:** es el componente más básico de una aplicación web y se encarga de dar significado al contenido y estructurarlo. Las "marcas" y "elementos" especiales son los que etiquetan el contenido de la web para mostrarlo. Utiliza etiquetas para distinguir a los elementos HTML[18].
- **CSS:** es un lenguaje de estilos que se encarga de la presentación de documentos HTML o XML. Su sintaxis permite renderizar los elementos de la aplicación web tomando en cuenta una serie de parámetros que establece el desarrollador, como lo pueden ser fuentes, colores, tamaños, animaciones y otras características [19].
- **JavaScript:** JavaScript es un lenguaje de programación interpretado con soporte para programación orientada a objetos, imperativa y declarativa[20]. Es un lenguaje que se suele usar en aplicaciones web para complementar el uso de HTML y CSS. Que sea un lenguaje interpretado significa que su lenguaje se

convierte a lenguaje de máquina a medida que se ejecuta el código de JavaScript, lo que lo hace más eficiente frente a otros lenguajes de programación compilados [21].

- **React:** es una librería de JavaScript utilizada para construir interfaces de usuario en el desarrollo de aplicaciones. Se caracteriza por el uso "componentes", que son fragmentos de código que crean diferentes partes de la interfaz y que luego se colocan unos con otros para armar la interfaz en su totalidad. Permite una sintaxis diferente y más amplia que la de JavaScript, ya que en React se puede utilizar lenguaje JavaScript y HTML en un mismo fichero. A esta sintaxis se le conoce como "JSX"[22]. Se complementa su desempeño con algunas otras librerías como *react-router* o *react-bootstrap*, utilizadas para el manejo de rutas del *frontend* y para poder usar los elementos de Bootstrap pero adaptados a React.
- **ChartJS:** es una librería gratuita de JavaScript que permite crear gráficos basados en HTML. Posee diferentes tipos de gráficos como de linea, de barras, entre otros [23].

2.1.1.3. Controlador

- **Python:** es un lenguaje de programación de código abierto con un enfoque multiparadigma, lo que quiere decir que es capaz de soportar diferentes tipos de porgramacion como la orientada a objetos, la procedural y la funcional [24]. Al igual que JavaScript, es un lenguaje interpretado [21]. En el caso de la aplicación que se expone en este TFG, se ha utilizado Python para conectarse a una base de datos MongoDB y procesar dichos datos. Este lenguaje es capaz de conectarse a la base de datos debido al uso del controlador (*driver*) oficial de MongoDB para aplicaciones síncronas desarrolladas con Python, denominado **PyMongo** [25].
- **Flask:** es un *framework* de Python orientado a crear programas que se ejecutan en el lado del servidor. Dichos programas ayudan al servidor web a manejar tanto las peticiones que realizan los navegadores como las respuestas a esas peticiones [26].

2.1.2. Funcionamiento

La arquitectura basada en el patrón MVC explicada anteriormente se relaciona con las dos partes fundamentales en las que se divide el desarrollo de la aplicación web, el *frontend* y el *backend* (ver Figura 2.2).

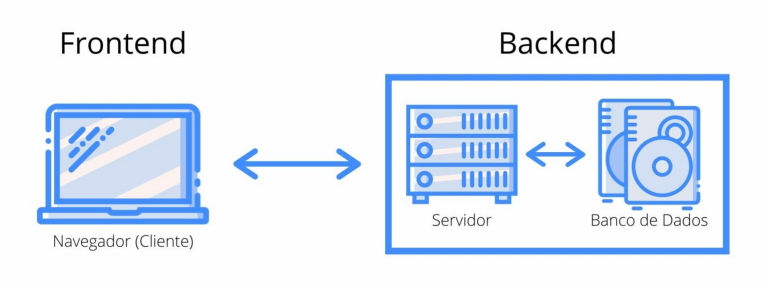


Figura 2.2: Esquema del flujo web entre el *frontend*, *backend* y una base de datos (tomado de: [27]).

A continuación se definen los conceptos de cada una de las partes de la aplicación, se mencionan las tecnologías que se han usado para su desarrollo y se explica como funciona la comunicación entre ellas, asi como la comunicación del *backend* con la base de datos. Además, se establece la relación de estas partes de la aplicación con los elementos del patrón MVC explicado anteriormente y se describen detalladamente las características de la base de datos a la que se conecta la aplicación y los datos utilizados por la misma.

2.1.2.1. Frontend

El *frontend* está formado por las tecnologías que corren en los navegadores, también conocidas como tecnologías del lado del cliente. Se centran en la experiencia de usuario, cómo este puede interactuar con la aplicación web y su visualización. Las tres tecnologías principales para el desarrollo de un *frontend* son: HTML, CSS y JavaScript [28].

Las tecnologías utilizadas para el desarrollo del *frontend* de la aplicación son: HTML, CSS, JavaScript y React, lo que indica que el *frontend* coincide con el elemento "Vista"del patrón MVC.

Las diferentes vistas que componen el *frontend* son:

- ***Vista de inicio de sesión (Log in):*** posee un formulario de acceso que funciona con un único usuario (“userNDT”) y una única clave (“NDT2023”). Al presionar en el botón de inicio de sesión, si las credenciale son correctas, se redirige al usuario a la vista inicial (*Home*). En versiones futuras deberá ser sustituido por un módulo de seguridad que se conecte a la base de datos (ver Figura 2.3).

Figura 2.3: vista inicio de sesión de la aplicación

- ***Vista inicial (Home):*** contiene la lista de los pacientes identificados por su "id" correspondiente en la base de datos. Cada paciente representa un desplegable que cuenta con cuatro botones para cada ejercicio que se visualizan al clicar sobre uno de ellos (ver Figura 2.4).

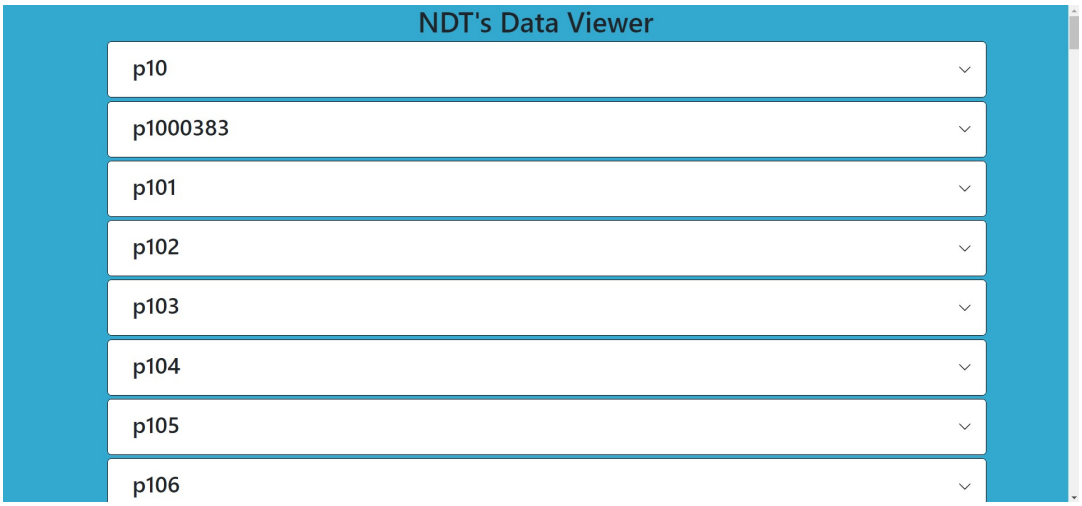


Figura 2.4: vista inicial de la aplicación

- ***Vista del ejercicio 1 (Flexo-extensión de muñeca):*** este ejercicio se basa en un movimiento repetitivo de las manos de abajo hacia arriba y toma como variables de interés el angulo de la mano respecto a la horizontal y el vector normal a la palma de la mano [29]. En esta vista se encuentran tres botones que corresponden a tres formas de visualizar los resultados de las variables mencionadas: graficar los resultados de la mano izquierda, graficar los resultados de la mano derecha o graficar ambas manos al mismo tiempo (ver Figura 2.5).

Figura 2.5: vista ejercicio 1

- ***Vista del ejercicio 2 (Pinza índice - pulgar):*** es un ejercicio que consiste en separar y juntar los dedos índice y pulgar. La habilidad para realizar la pinza y la manera en que se logra dicho movimiento son relevantes para la obtención de los resultados [29]. En este caso, la vista posee cuatro botones, cada uno correspondiente a una de las variables de interés: apertura máxima de pinza, rango del dedo pulgar, rango del dedo índice y el perímetro del dedo anular (ver Figura 2.6).

Figura 2.6: vista ejercicio 2

- ***Vista del ejercicio 3 (Separación de dedos):*** el movimiento requerido en este ejercicio se realiza sobre el plano x-z y se mide el rango de movmiento de los dedos a partir de la punta de los mismos [29]. Para esta vista se presentan cuatro botones para representar el rango de movimiento de cuatro dedos distintos: pulgar, índice, anular y meñique (ver Figura 2.7).

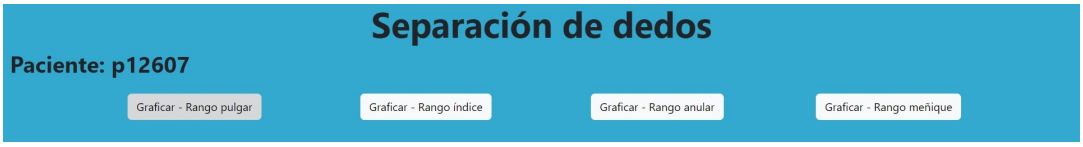


Figura 2.7: vista ejercicio 3

- ***Vista del ejercicio 4 (Apertura - cierre de puño):*** para el análisis de este movimiento se toman en cuenta la punta de todos los dedos en el plano x-z, donde se posicionan al abrir el puño [29]. En este ejercicio hay dos botones que representan el resultado de las variables de interés: perímetro máximo de cada mano y el rango de movimiento del pulgar de cada mano (ver Figura 2.8).

Figura 2.8: vista ejercicio 4

Las vistas de los ejercicios poseen el título del ejercicio, el identificador del paciente y diferentes botones que permiten visualizar las diferentes gráficas de cada uno de ellos. Por lo que la forma en que se presentan estas vistas puede variar por paciente y por gráfica elegida de cada ejercicio.

En el Capítulo 3 se comentarán, de manera más completa, las variables de interés de los ejercicios correspondientes a las vistas presentadas previamente.

2.1.2.2. Backend

El *backend* está formado por las tecnologías que ejercen sus funciones en el lado del servidor. Tienen diversos objetivos como interactuar con bases de datos, servir todas las vistas y otras actividades. Para el desarrollo de un *backend* existe una gran variedad de lenguajes de programación, entre ellos se encuentran PHP, Python, Java, entre otros [28].

Las tecnologías utilizadas para el desarrollo del *backend* de la aplicación son: Python y Flask, lo que indica que el *frontend* coincide con el elemento "Controlador" del patrón MVC.

2.1.2.3. Características de la base de datos y especificación de datos utilizados por la aplicación

La aplicación se conecta al servidor de ROBOLABO en donde se encuentran dos bases de datos no relacionales desarrolladas con MongoDB. Una de ellas posee una colección, mientras que la otra posee tres colecciones. Los datos que albergan se han obtenido de pruebas clínicas realizadas en el Hospital Universitario La Paz, específicamente en el áre de neurología, con un dispositivo médico llamado NeuroData Tracker y que constan de cuatro ejercicios por prueba [29]. Este dispositivo está orientado a la cuantificación del déficit motor en las manos de pacientes que han sufrido al menos un ictus [29]. Por lo que la información que se almacena en estas bases de datos son datos personales, antecedentes clínicos y los resultados de los ejercicios realizados en las pruebas clínicas de cada paciente.

La estructura de las bases de datos que se encuentran en el servidor del Laboratorio de Robótica y Control de la ETSIT-UPM es la siguiente (ver Figura 2.8):

- Base de datos 1 - *Database 1*

    - Colección *Usuarios*
- Base de datos 2 - *Database 2*
    - Colección *Antecedentes de ictus*
    - Colección *Datos clínicos*
    - Colección *Estudios*

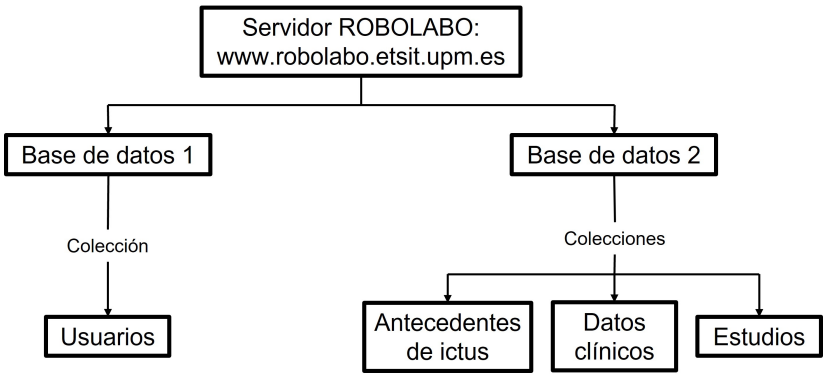


Figura 2.9: Estructura de las bases de datos alojadas en el servidor de ROBOLABO

La aplicación se centra en el uso de los datos de los ejercicios de cada paciente, además otros datos como el *id* y el usuario de cada uno de los pacientes. Es importante comentar que los datos de los ejercicios son variables cinemáticas del movimiento de las manos de los pacientes y sus valores numéricos están guardados en formato tipo *string* en los documentos que componen la base de datos, es decir, todas las variables medidas en los cuatro ejercicios realizados por los pacientes se guardan como una larga cadena de texto, por lo que la aplicación se encarga de hacer un procesado de datos, que se explica detalladamente más adelante, antes de graficar las variables de interés en el *frontend*.

En cada ejercicio se miden 24 variables cinemáticas por cada mano y el tiempo de cada ejercicio, que siempre son un total de cinco segundos. Como en cada ejercicio se emplean ambas manos, se tienen un total de 48 variables cinemáticas del movimiento de las manos y una variable con el tiempo del ejercicio.

2.1.2.4. Comunicación entre frontend y backend

Para entender cómo se comunican las dos partes que componen la aplicación, primero se debe establecer el concepto de API y del formato de datos JSON, así como los diferentes métodos de peticiones HTTP.

Una API (*Application programming interface*) es un programa que se encarga de canalizar la información de una parte de un software a otra. Cumple un papel de intermediario que permite extraer datos de un software y se utilicen en otro nivel del mismo o, incluso, en otro programa [30].

Por otro lado, JSON (JavaScript Object Notation) "es un formato ligero de intercambio de datos" [31]. Es completamente independiente del lenguaje JavaScript pero utiliza convenios familiares para los programadores, por lo que es ideal para el intercambio de datos [31].

En este caso, la aplicación utiliza una API de JavaScript denominada "Fetch API". Esta API proporciona una interfaz para recuperar recursos que funciona, incluso, a través de la red [32]. Su método asíncrono *fetch()* toma como argumento obligatorio la ruta de acceso al recurso que se quiere recuperar y es capaz de manejar las peticiones y respuestas relacionadas a dicha dirección [33].

Además, se utiliza otro método, denominado *json()*, para que la respuesta obtenida a la petición sea en formato JSON y sea legible para las tecnologías que componen el *frontend* de la aplicación. En caso de no utilizarlo, la respuesta de la petición sería una respuesta HTTP y no un archivo JSON [33].

Las peticiones HTTP son el medio de intercambio de datos entre servidores y clientes [34]. con la API Fetch pueden ser de diferentes tipos dependiendo del método HTTP que se utilice en dicha petición. Los posibles métodos de peticiones HTTP son [35]:

- **GET:** solicita una representación del recurso deseado. Es un método que solo recupera información.
- **HEAD:** esta solicitud obtiene una respuesta identica al método anterior pero sin el cuerpo de la respuesta. Es decir, solo se obtienen las cabeceras de la respuesta.
- **POST:** con este método se puede cargar un elemento al recurso y causar cambios de estado o efectos colaterales en el servidor.
- **PUT:** crea nuevos recursos o los reemplaza en la dirección de destino.
- **DELETE:** elimina recursos especificados.
- **CONNECT:** establece un tunel al servidor destino. Es una comunicación bidireccional.
- **OPTIONS:** describe opciones de comunicación.
- **TRACE:** envía un mensaje de prueba desde el origen hasta la dirección especificada.
- **PATCH:** aplica modificaciones parciales a un recurso.

En la aplicación se utilizan dos de estos métodos de solicitudes HTTP, el método "GET" y el método "POST", ambos utilizados en componentes del *frontend* para comunicarse con el *backend* y que este le proporcione la información recuperada, enformato JSON, de la base de datos. En el apartado de resultados se profundizará en los componentes que hacen las peticiones y que respuestas se obtienen.

2.1.2.5. Comunicación entre backend y la base de datos

El *backend* de la aplicación de visualización de datos se conecta a la base de datos descrita previamente. Esto se logra mediante el lenguaje Python, con el que se ha generado un archivo, perteneciente al *backend*, llamado “ConnectionDB.py” y que contiene una función que recibe dos parámetros numéricos. Dicha función utiliza la url del servidor y los dos parámetros numéricos para saber el nombre de la base de datos y la colección a la que debe acceder. Esta función es utilizada por otros ficheros del *backend* al momento del procesado de datos.

El *backend* cuenta con dos archivos por cada ejercicio, resultando en un total de 8 archivos para la recuperación y procesado de los datos que posteriormente se utilizan en el *frontend*. Cada ejercicio cuenta con un archivo que contiene una función que accede a la base de datos (usando la función del archivo ConnectionDB), recupera la información deseada, realiza una parte del procesado de datos y devuelve el resultado. Dicho resultado es utilizado en el otro archivo que se encarga del procesado de datos restante para obtener los valores de las variables de interés. Cabe mencionar que ambos archivos reciben un parámetro que utilizan para buscar al paciente que se desea en la base de datos. Dicho parámetro es el usuario asociado al paciente en la base de datos que contiene los resultados de los ejercicios

Se debe tener en cuenta que el *backend* cuenta con el uso de la tecnología Flask, explicada anteriormente, para crear un programa que ayuda a la aplicación y al servidor a obtener la información de la base de datos. Esto se logra mediante un archivo llamado "server.py"que al ejecutarse crea una serie de rutas virtuales en el *backend*, específicamente un total de cinco rutas. Una de ellas recupera los nombres y usuarios de los pacientes, mientras que las otras cuatro rutas recuperan los datos de cada ejercicio, siendo una ruta para cada ejercicio. Estas son las rutas a las que accede el *frontend* para recuperar los datos que se muestran en las diferentes vistas.

2.1.2.6. Procesado de datos

El *backend* posee la capacidad de hacer un procesado de datos que se divide en dos partes para cada ejercicio. Por ello, como se menciona anteriormente, por cada ejercicio hay dos archivos desarrollados en Python, uno que recupera los datos y hace parte del procesado, mientras que el otro archivo termina ese procesado de datos y otorga como respuesta las variables de interés del ejercicio. Este procesado se ejecuta de manera individual por paciente, ya que los archivos destinados a esta tarea contienen funciones que reciben como parámetro el usuario del paciente para poder hacer la búsqueda en la base de datos y solo recuperar los valores de los ejercicios de dicho paciente, lo que evita que se manejen muchos datos de varios pacientes al mismo tiempo.

Las etapas del procesado de datos se describen a continuación:

1. **Primera parte - Recuperación y su preparación:** en esta parte del procesado se utiliza un archivo que se conecta a la base de datos y recupera la información que contiene las variables cinemáticas de cada ejercicio. Como estas se encuentran guardadas como una larga cadena de texto, se realizan los siguientes pasos para obtener un diccionario de valores que será utilizado en la segunda parte del procesado:
   - Se separan por espacios en blanco el *string* que contiene los valores.
   - Se vuelve a separar el string por "punto y coma"(;), pero solamente la primera fila, ya que es utilizada para crear las claves del diccionario, ya que contiene los nombres de las variables.
   - Se vuelve a separar el string por "punto y coma"(;) pero esta vez a partir de la fila 1 (la fila 0 pertenece al paso anterior). De esta forma se obtiene que cada variable del paso anterior se asocia con los valores debidos.
   - Se devuelve como respuesta un diccionario que contiene todas las variables medidas por el dispositivo.

   Las variables medidas por el dispositivo que se colocan en el diccionario de valores devuelto por la primera parte del procesado de datos se muestran en la Tabla 2.1:

Variables cinemáticas		
	Mano izquierda	Mano derecha
Posición de la punta de los dedos	leftThumbTipPosition_X leftThumbTipPosition_Y leftThumbTipPosition_Z leftIndexTipPosition_X leftIndexTipPosition_Y leftIndexTipPosition_Z leftMiddleTipPosition_X leftMiddleTipPosition_Y leftMiddleTipPosition_Z leftRingTipPosition_X leftRingTipPosition_Y leftRingTipPosition_Z leftPinkyTipPosition_X leftPinkyTipPosition_Y leftPinkyTipPosition_Z	rightThumbTipPosition_X rightThumbTipPosition_Y rightThumbTipPosition_Z rightIndexTipPosition_X rightIndexTipPosition_Y rightIndexTipPosition_Z rightMiddleTipPosition_X rightMiddleTipPosition_Y rightMiddleTipPosition_Z rightRingTipPosition_X rightRingTipPosition_Y rightRingTipPosition_Z rightPinkyTipPosition_X rightPinkyTipPosition_Y rightPinkyTipPosition_Z
Posición de la palma	leftPalmPosition_X leftPalmPosition_Y leftPalmPosition_Z	rightPalmPosition_X rightPalmPosition_Y rightPalmPosition_Z
Velocidad de la palma	leftPalmVelocity_X leftPalmVelocity_Y leftPalmVelocity_Z	rightPalmVelocity_X rightPalmVelocity_Y rightPalmVelocity_Z
Vector normal de la palma	leftPalmNormal_X leftPalmNormal_Y leftPalmNormal_Z	rightPalmNormal_X rightPalmNormal_Y rightPalmNormal_Z

Tabla 2.1: Variables cinemáticas obtenidas con el dispositivo óptico de captura de movimiento [29].

A continuación se explican las variables de la Tabla 2.1:

- **Las variables cuyo nombre termina con "Position_X", "Position_Y" y "Position_Z":** son variables que miden la posición de la parte de la mano respectiva, que puede ser cualquiera de los dedos o la palma de la mano. El propio nombre, en inglés, de cada variable indica si se refiere a la palma de la mano o alguno de los dedos, además de que también indica si se está hablando de la mano izquierda o derecha. Por último, la letra al final del nombre de los archivos denota el eje en el que se ha medido la componente del vector de dicha variable (Primera y segunda fila de la Tabla 2.1). (Primera y segunda fila de la Tabla 2.1)
- **Las variables cuyo nombre termina con "Velocity_X", "Velocity_Y" y "Velocity_Z":** son variables que solo miden en la palma de la mano, ya sea izquierda o derecha, y se refiere a la velocidad de dicha parte

durante el movimiento realizado en el ejercicio. El nombre completo de los archivos, en inglés, también denota la mano de la que se está midiendo la variable. Por último, la letra al final del nombre de los archivos denota el eje en el que se ha medido la componente del vector de dicha variable (Tercera fila de la Tabla 2.1).

- **Las variables cuyo nombre termina con "Normal_X", "Normal_Y" y "Normal_Z":** son varaibles que miden el vector normal a la palma de la mano, por ello no hay ninguna variable de este tipo que haga referencia a los dedos. Por último, la letra al final del nombre de los archivos denota el eje en el que se ha medido la componente del vector de dicha variable.

2. **Segunda parte - operaciones matemáticas:** en esta parte del procesado se toma el diccionaro devuelto por la función del archivo mencionado en la parte anterior y se realizan una serie de operaciones matemáticas, que varían dependiendo del ejercicio para obtener las variables de interés de cada ejercicio y devolverlas dentro de un diccionario de valores. Esta es la respuesta que se envía a las direcciones virtuales del *backend* y que son recuperadas por las peticiones del *frontend*.

Capítulo 3

Resultados de la aplicación web de visualización de datos

Para el desarrollo de este capítulo se debe tener en cuenta lo mencionado en el capítulo anterior acerca del origen de los datos, ya que se parte de información obtenida en pruebas clínicas realizadas en el Hospital Universitario de La Paz con el dispositivo NeuroData Tracker. Se debe destacar que los datos que se grafican en la aplicación son variables cinemáticas del movimiento de las manos de los pacientes y que se obtienen tras procesar los datos iniciales medidos por el dispositivo.

Para presentar los resultados de la aplicación, este capítulo se divide en dos partes:

- **Visualización de datos - primera parte:** se exponen las gráficas obtenidas en cada ejercicio junto con breves comentarios descriptivos sobre la misma.
- **Prestaciones de la aplicación - segunda parte:** se describen tiempos de carga de las gráficas, de las diferentes vistas y de la propia aplicación web.

3.1. Visualización de datos

Es preciso recordar que cada ejercicio posee un rango de normalidad para cada mano, es decir, un rango de valores que indican si el resultado obtenido por el paciente se encuentra entre los valores que son usualmente obtenidos en personas que no han presentado patologías que les impidan realizar los movimientos de cada ejercicio. Estos rangos de normalidad se calcularon en un estudio clínico, perteneciente al proyecto del dispositivo NeuroData Tracker [29] realizado en el Hospital Universitario La Paz. Dichos valores provienen de los resultados obtenidos en cada ejercicio, y en cada mano, de los controles (estudios en personas sanas).

Los rangos de normalidad establecidos para cada ejercicio se grafican con una línea gris. Se pueden presentar tres casos que se describen a continuación:

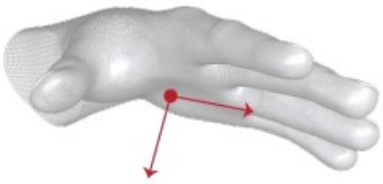
- **El resultado está dentro del rango de normalidad:** cuando el resultado está comprendido entre el límite superior e inferior del rango de normalidad, el valor de dicho resultado se mostrará en color verde, indicando que es un resultado normal.

- **El resultado es inferior al rango de normalidad:** cuando el resultado es menor que el límite inferior del rango de normalidad, el valor de dicho resultado se mostrará en color rojo, indicando que es un resultado que no es normal.
- **El resultado es superior al rango de normalidad:** cuando el resultado es mayor que el límite superior del rango de normalidad, el valor de dicho resultado se mostrará en color azul, indicando que es un resultado fuera de los rangos de normalidad pero no supone una minoración en el movimiento del ejercicio. El único ejercicio en el que no se cumple esta regla es en el ejercicio 2, concretamente a la hora de graficar el perímetro del dedo anular, ya que en este caso se grafica de color rojo al superar el rango de normalidad.

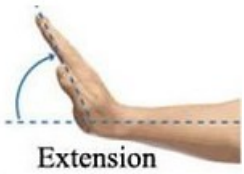
Los resultados gráficos y valores que se muestran a continuación toman como ejemplo los datos obtenidos en las pruebas realizadas al paciente **p12607**, identificado mediante el campo "_id" de la base de datos 1 y la colección *Usuarios*

3.1.1. Ejercicio 1: flexo - Extensión de muñeca

Este ejercicio se basa en un movimiento de arriba a abajo de las manos en el que se tienen en cuenta dos variables de interés: el vector normal a la palma de la mano y el ángulo de la mano con respecto a la horizontal.



(a) Vector normal a la palma de la mano [36]



(b) Ángulo de la mano con la horizontal [37]

En la Figura 3.1 se pueden observar los resultados obtenidos para la mano izquierda, mientras que en la Figura 3.2 se tienen los resultados de la mano derecha. En ambos casos, a la izquierda se presenta una gráfica que representa la evolución de la componente "Z" del vector normal a la mano correspondiente, frente al tiempo. A la derecha se presenta una gráfica donde se indica el máximo valor del ángulo del vector normal con respecto a la horizontal (en grados).

Figura 3.1: Resultados del ejercicio 1 - Mano izquierda

Figura 3.2: Resultados del ejercicio 1 - Mano derecha

Desde la Figura 3.3 a la Figura 3.5 se muestran tres partes que conforman los resultados de ambas manos. Es una opción en la vista del ejercicio 1 para que se puedan visualizar al mismo tiempo.

La primera parte de esta modalidad (Figura 3.3) de visualización consta de la gráfica que representa la evolución del vector normal a la palma de la mano izquierda frente al tiempo. Para la segunda parte (Figura 3.4) se tiene la misma gráfica pero para la mano derecha. Por último, se muestra una gráfica (Figura 3.5) con los máximos ángulos respecto a la horizontal, de ambas manos, que se miden en grados.

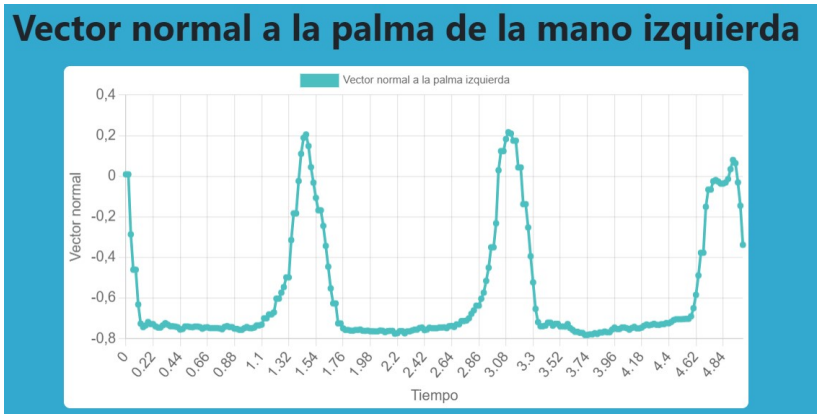


Figura 3.3: Resultados del ejercicio 1 - Ambas manos (Pt.1)

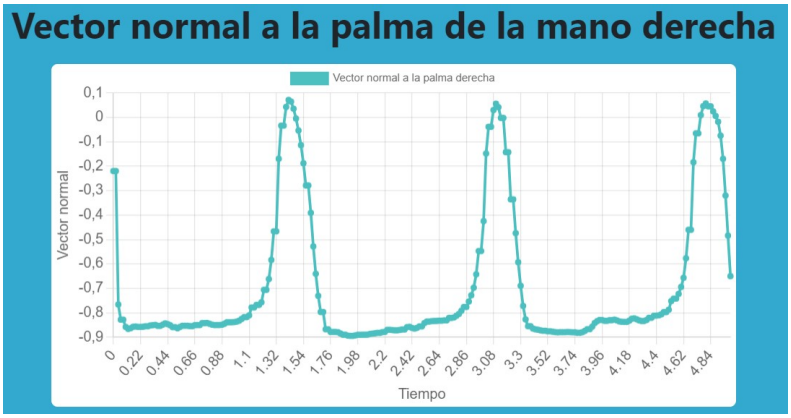


Figura 3.4: Resultados del ejercicio 2 - Ambas manos (Pt.2)

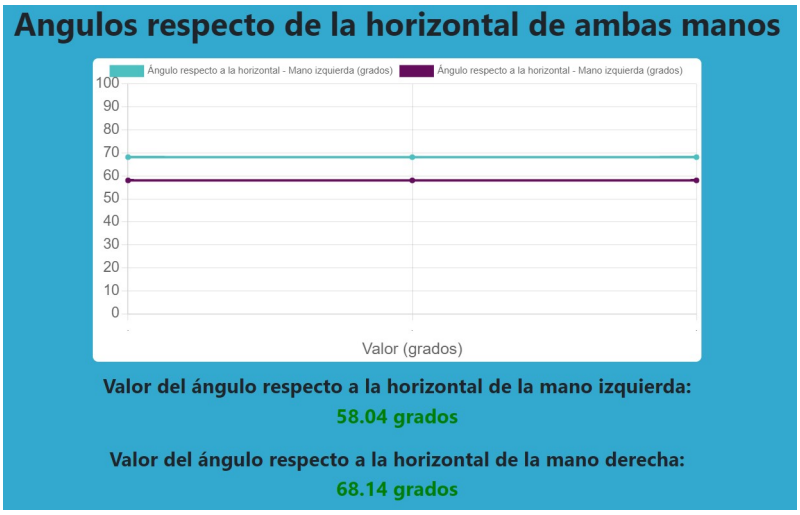


Figura 3.5: Resultados del ejercicio 1 - Ambas manos (Pt.3)

En la última gráfica (Figura 3.5), como se grafican los valores de ambas manos, se optó por no representar los rangos de normalidad ya que cada mano posee valores diferentes y se podrían crear confusiones respecto a los resultados obtenidos. Lo que se busca con esta gráfica es, únicamente, la comparación entre los ángulos de ambas manos. En caso de que una mano tenga un valor significativamente más alto puede ser un indicador útil a la hora de la interpretación de los datos

3.1.2. Ejercicio 2: pinza índice - pulgar

A partir de este ejercicio se presenta una modalidad de visualización en la que la gráfica de la izquierda siempre representa los resultados de la mano izquierda y la gráfica de la derecha representa los resultados de la mano derecha. Esto se cumple para las variables de interés tanto del ejercicio actual (ejercicio 2) como del ejercicio 3 y 4.

En este ejercicio 2 se tienen cuatro variables de interes calculadas para ambas manos: apertura máxima de pinza, rango máximo del dedo pulgar, rango máximo del dedo índice y el perímetro del dedo anular (ver Figura 3.6). Cabe mencionar que, en este caso, todas ellas se miden en milímetros.

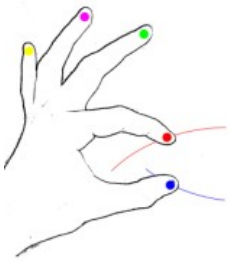


Figura 3.6: Apertura de pinza, rango máximo del pulgar y del índice. Perímetro de movimiento anular [38]

En la Figura 3.7 se encuentran los resultados obtenidos para la primera variable de interés, la apertura máxima de pinza. Es la distancia que se crea entre el dedo índice y pulgar al momento de separar dichos dedos.

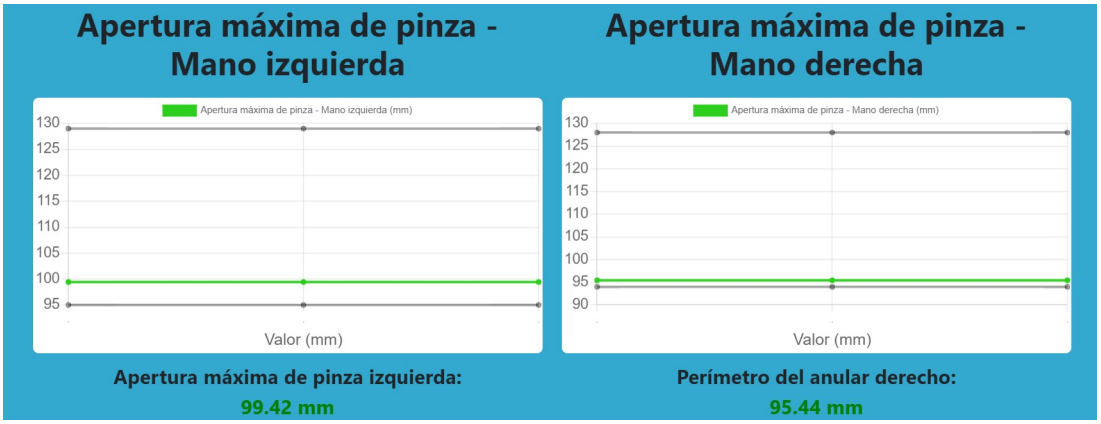


Figura 3.7: Resultados del ejercicio 2 - Apertura máxima de pinza

En la Figura 3.8 se representa la segunda variable de interés, el rango máximo del pulgar. Este valor representa la distancia máxima recorrida por el pulgar en una apertura, que no necesariamente debe coincidir con la apertura máxima.

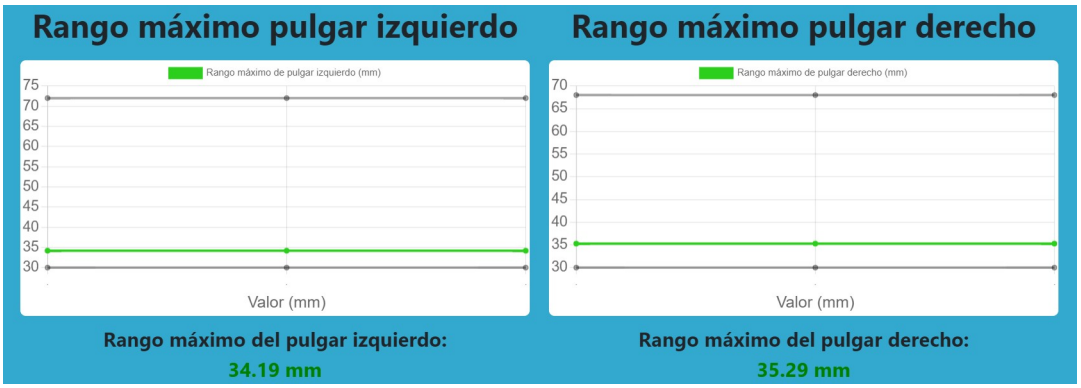


Figura 3.8: Resultados del ejercicio 2 - Rango máximo del pulgar

En la Figura 3.9 se representa la misma medida que en la figura anterior pero para el dedo índice. Se mide la distancia máxima recorrida por dicho dedo en una apertura que puede, o no, coincidir con la apertura máxima de la pinza.

Figura 3.9: Resultados del ejercicio 2 - Rango máximo del índice

En la Figura 3.10 se observan los resultados obtenidos para la última variable de interés del ejercicio 2, el perímetro del dedo anular. Esta medida se utiliza para saber qué movimiento hay en dicho dedo mientras se realiza la pinza.

Figura 3.10: Resultados del ejercicio 2 - Perímetro del anular

3.1.3. Ejercicio 3: separación de dedos

Al igual que en el ejercicio 2, este ejercicio cuenta con cuatro variables de interés: rango máximo de los dedos pulgar, índice, anular y meñique. Se toman en cuenta estos cuatro dedos ya que son los que tienden a presentar mayor movimiento durante la ejecución de este movimiento (ver Figura 3.11).

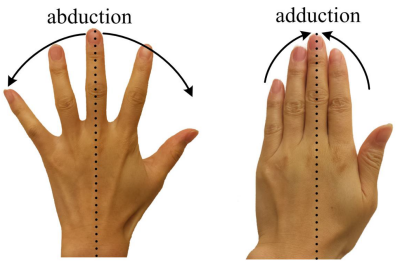


Figura 3.11: Separación de dedos - Rango de movimiento de los dedos involucrados [39]

Como se ha comentado anteriormente, el rango máximo de un dedo es la distancia máxima recorrida por dicho dedo durante la realización del ejercicio. No se debe confundir este concepto con la distancia total recorrida que se refiere a la suma de todos los desplazamientos del dedo hasta el fin del ejercicio. Por otro lado, el valor que se representa como rango máximo de un dedo es uno de los varios desplazamientos del dedo en donde alcanzó la mayor distancia recorrida durante el ejercicio.

Figura 3.12: Resultados del ejercicio 3 - Rango máximo pulgar

Figura 3.13: Resultados del ejercicio 3 - Rango máximo índice

Figura 3.14: Resultados del ejercicio 3 - Rango máximo anular

Figura 3.15: Resultados del ejercicio 3 - Rango máximo meñique

3.1.4. Ejercicio 4: apertura - cierre de puño

En el último ejercicio el paciente debe intentar abrir y cerrar el puño. En este caso se calculan dos variables de interés: el perímetro máximo de cada mano y el rango máximo del pulgar (ver Figura 3.16).

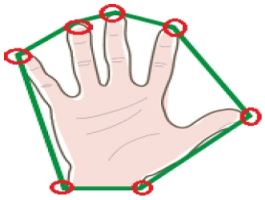


Figura 3.16: Perímetro máximo de la mano (verde) y rango máximo del pulgar al abrir el puño [40]

En esta primera gráfica (Figura 3.17) se representa el perímetro máximo obtenido para cada mano.

Figura 3.17: Resultados del ejercicio 4 - Perímetro máximo mano izquierda

En la segunda y última gráfica (Figura 3.18) se representa el rango máximo del dedo pulgar, por lo que se debe recordar que por ser un rango máximo no se debe confundir con el concepto de distancia total, mencionado previamente.

Figura 3.18: Resultados del ejercicio 4 - Rango máximo pulgar

3.2. Tiempos de carga del contenido gráfico de la aplicación

Para medir los tiempos de carga de las diferentes gráficas que componen a cada ejercicio, se utilizaron las herramientas de desarrollador del navegador *Google Chrome*, que se activan presionando la tecla **F12** mientras se está en la vista del ejercicio seleccionado.

Al abrir las herramientas de desarrollador se encuentra un panel de control con diferentes secciones, dentro de las cuales está la sección "Red". Dicha sección se encarga de registrar las diferentes peticiones que se realizan al interactuar, en este caso, con la interfaz de la aplicación.

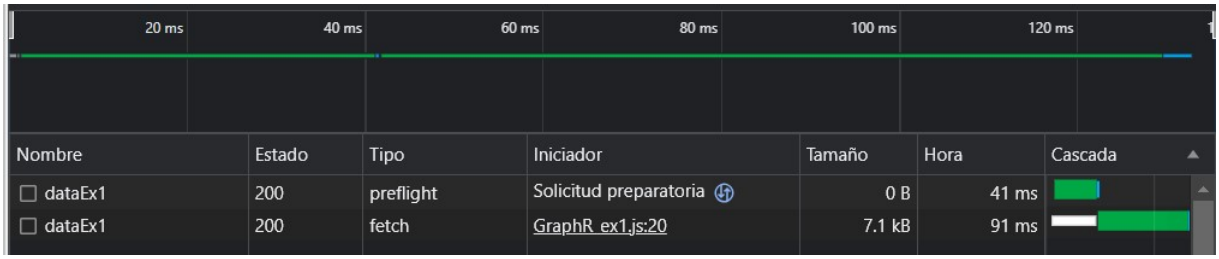


Figura 3.19: Herramientas de desarrollador y su sección "Red" luego de realizar una petición con la aplicación

Como se muestra en la Figura 3.19, las peticiones que realiza la aplicación se componen de dos partes:

- **La solicitud preparatoria:** petición que comprueba que el protocolo CORS está permitido [41].
- **La solicitud con el método *fetch()*:** es la solicitud que recupera los datos del *backend* que se grafican en la aplicación.

Para medir los tiempos de carga de cada gráfica de cada ejercicio se hicieron pruebas de carga con cinco pacientes al azar, calculando el tiempo de carga como la suma del tiempo en que se realiza tanto la solicitud preparatoria como la petición del método *fetch()* para cada gráfica. Con dichos datos se calcularon medias y desviaciones típicas para hacer una estimación del tiempo de carga que le corresponde a cada gráfica. Además de la representación gráfica de los resultados de las pruebas en diagramas de caja.

Se debe especificar que dichas pruebas se ejecutaron en un ordenador portatil con procesador Intel Core i7, 16 GB de memoria RAM, 237 GB de almacenamiento interno y un discoduro de 1 TB, con las siguientes condiciones de red:

- Velocidad de subida: 80.70 Mb/s. Es la velocidad de descarga de datos [42].
- Velocidad de bajada: 35.24 Mb/s. Es la velocidad en que se envía la información desde un dispostivo a Internet [42].
- Latencia: 18 ms. Es el tiempo que tarda en viajar un paquete desde que el navegador realiza la petición hasta que llega la respuesta [43].

Estos parámetros se han aclarado para contextualizar las pruebas de carga realizadas, debido a que si se repiten las mismas pruebas en un dispostivo diferente y/o condiciones de red distintas los resultados pueden variar.

Para cada ejercicio se han obtenido:

- Tabla con los tiempos de carga, en milisegundos, de cada gráfica para cada paciente. Además de la media y la desviación típica de dichos valores.
- Tabla con los valores temporales, en milisegundos, asociados a su respectiva gráfica.

- Diagrama de "cajas y bigotes" para evaluar la variablidad de los datos de tiempo de carga.

A continuación se presentan los resultados de las pruebas de carga para cada ejercicio:

- **Tiempos de carga para gráficas del ejercicio 1:** la Figura 3.21 muestra la tabla de los tiempos de carga que se obtuvieron en las pruebas con los cinco pacientes. Además, contiene el valor de la media de dichos tiempos de carga y su desviación típica para el ejercicio 1.

	Tiempos de carga de gráficas del ejercicio 1		
Paciente	Mano Izquierda (ms)	Mano Derecha (ms)	Ambas manos (ms)
P10	184	139	122
P105	119	148	108
P12607	126	173	117
P114	161	129	132
P262219	133	124	221
Media	144,6	142,6	140
Desviación típica	27,19007172	19,34683437	46,10314523

Figura 3.20: Tabla con los tiempos de cargas de cada gráfica del ejercicio 1 correspondientes a cada paciente, la media y la desviación típica.

La Figura 3.22 muestra la tabla compuesta por las gráficas del ejercicio 1 y los cinco valores de tiempo de carga que se obtuvieron para los cinco pacientes. Es el origen del diagrama de caja y bigotes de la Figura 3.23.

Gráficas	Tiempos (ms)
Mano Izquierda (ms)	184
Mano Izquierda (ms)	119
Mano Izquierda (ms)	126
Mano Izquierda (ms)	161
Mano Izquierda (ms)	133
Mano Derecha (ms)	139
Mano Derecha (ms)	148
Mano Derecha (ms)	173
Mano Derecha (ms)	129
Mano Derecha (ms)	124
Ambas manos (ms)	122
Ambas manos (ms)	108
Ambas manos (ms)	117
Ambas manos (ms)	132
Ambas manos (ms)	221

Figura 3.21: Tabla con todos los valores temporales de cada gráfica del ejercicio 1

La Figura 3.23 contiene el diagrama de caja y bigotes generado a partir de la

tabla de la Figura 3.22. En dicho diagrama se puede observar como la petición que grafica el resultado de "Ambas manos" es el que presenta mayor variabilidad en sus datos, mientras que el resultado de "Mano Derecha" es el que presenta menor variabilidad.

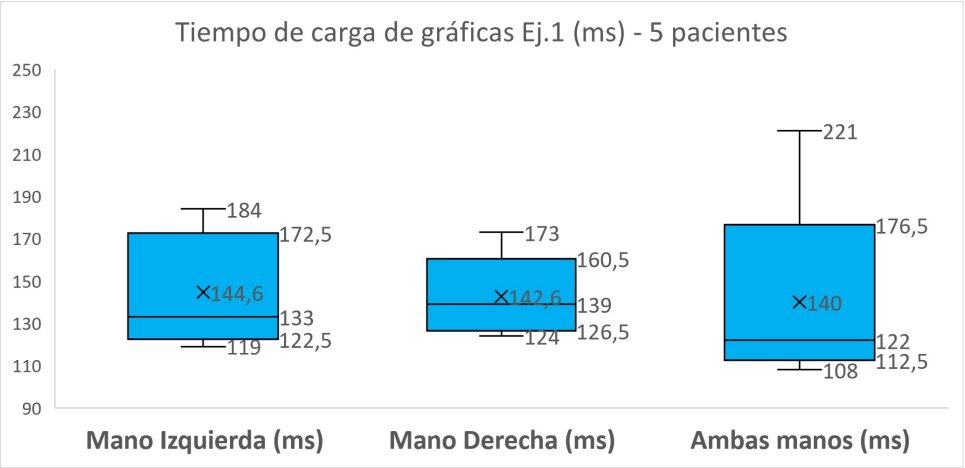


Figura 3.22: Diagrama de caja y bigotes de los tiempos de carga del ejercicio 1

Para el ejercicio 1 se obtuvieron los siguientes tiempos de carga medios y desviaciones típicas:

- Gráfica *Mano Izquierda*:
  - Tiempo de carga medio = 144,6 ms
  - Desviación típica = 27,19 ms
- Gráfica *Mano Derecha*:
  - Tiempo de carga medio = 142,6 ms
  - Desviación típica = 19,35 ms
- Gráfica *Ambas manos*:
  - Tiempo de carga medio = 140 ms
  - Desviación típica = 46,1 ms

Al comparar dichos resultados con el diagrama de caja y bigotes los valores de media coinciden, además de que la mayor desviación típica la posee la gráfica de ambas manos, lo que concuerda con el gran tamaño de su caja correspondiente en el diagrama de la Figura 3.23.

- **Tiempos de carga para gráficas del ejercicio 2:** la Figura 3.24 muestra la tabla de los tiempos de carga que se obtuvieron en las pruebas con los cinco pacientes. Además, contiene el valor de la media de dichos tiempos de carga y su desviación típica para el ejercicio 2.

	Tiempos de carga de gráficas del ejercicio 2			
Paciente	Apertura pinza (ms)	Rango pulgar (ms)	Rango índice (ms)	Perímetro anular (ms)
P10	148	107	134	123
P105	123	119	186	112
P12607	131	164	136	126
P114	119	124	124	109
P262219	114	126	123	119
Media	127	128	140,6	117,8
Desviación típica	13,28533026	21,43595111	26,03459237	7,190271205

Figura 3.23: Tabla con los tiempos de cargas de cada gráfica del ejercicio 2 correspondientes a cada paciente, la media y la desviación típica.

La Figura 3.25 muestra la tabla compuesta por las gráficas del ejercicio 2 y los cinco valores de tiempo de carga que se obtuvieron para los cinco pacientes. Es el origen del diagrama de caja y bigotes de la Figura 3.26.

Gráficas	Tiempos
Apertura pinza (ms)	148
Apertura pinza (ms)	123
Apertura pinza (ms)	131
Apertura pinza (ms)	119
Apertura pinza (ms)	114
Rango pulgar (ms)	107
Rango pulgar (ms)	119
Rango pulgar (ms)	164
Rango pulgar (ms)	124
Rango pulgar (ms)	126
Rango indice (ms)	134
Rango indice (ms)	186
Rango indice (ms)	136
Rango indice (ms)	124
Rango indice (ms)	123
Perimetro anular (ms)	123
Perimetro anular (ms)	112
Perimetro anular (ms)	126
Perimetro anular (ms)	109
Perimetro anular (ms)	119

Figura 3.24: Tabla con todos los valores temporales de cada gráfica del ejercicio 2

La Figura 3.26 contiene el diagrama de caja y bigotes generado a partir de la tabla de la Figura 3.25. En dicho diagrama se puede observar como la petición que grafica el resultado de “Rango índice” es el que presenta mayor variabilidad en sus datos, mientras que el resultado de “Perímetro anular” es el que presenta menor variabilidad.

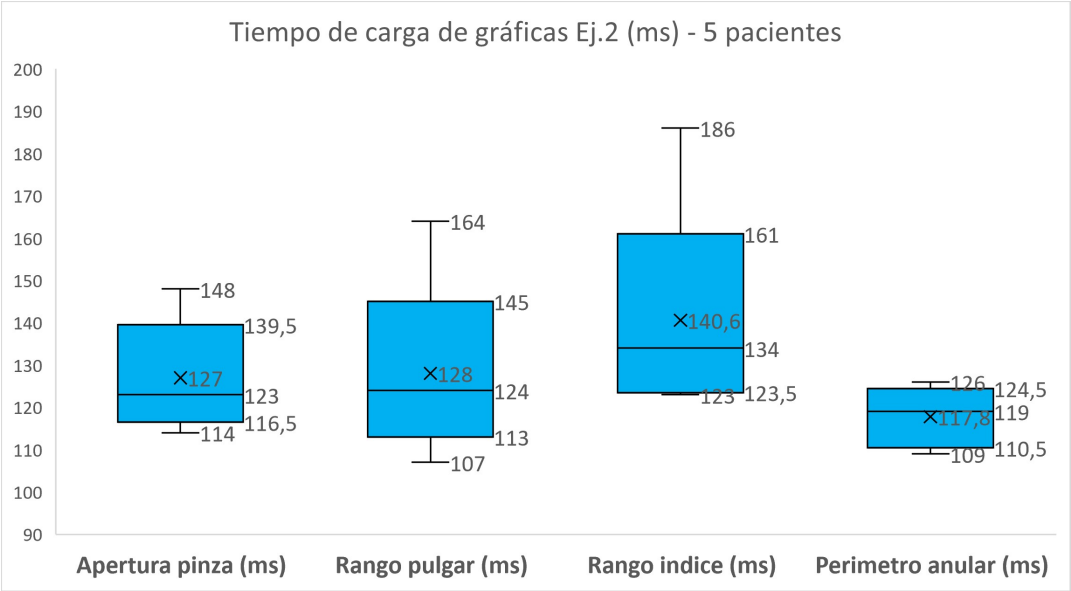


Figura 3.25: Diagrama de caja y bigotes de los tiempos de carga del ejercicio 2

Para el ejercicio 2 se obtuvieron los siguientes tiempos de carga medios y desviaciones típicas:

- Gráfica *Apertura pinza*:
  - Tiempo de carga medio = 127 ms
  - Desviación típica = 13,29 ms
- Gráfica *Rango pulgar*:
  - Tiempo de carga medio = 128 ms
  - Desviación típica = 21,44 ms
- Gráfica *Rango índice*:
  - Tiempo de carga medio = 140,6 ms
  - Desviación típica = 26,03 ms
- Gráfica *Perímetro anular*:
  - Tiempo de carga medio = 117,8 ms
  - Desviación típica = 7,19 ms

Al comparar dichos resultados con el diagrama de caja y bigotes los valores de media coinciden, al igual que el tamaño de las cajas respecto a los valores de las desviación típica. La mayor desviación típica la posee la graficación del rango del dedo índice, lo que concuerda con el gran tamaño de su caja correspondiente en el diagrama de la Figura 3.26. Además, la graficación del períemtro del dedo anular tiene una desviación típica muy pequeña por lo que su valor de tiempo de carga es muy similar cada vez que se realiza la petición.

- **Tiempos de carga para gráficas del ejercicio 3:** la Figura 3.27 muestra la tabla de los tiempos de carga que se obtuvieron en las pruebas con los cinco pacientes. Además, contiene el valor de la media de dichos tiempos de carga y su desviación típica para el ejercicio 3.

	Tiempos de carga de gráficas del ejercicio 3			
Paciente	Rango pulgar (ms)	Rango índice (ms)	Rango anular (ms)	Rango meñique (ms)
P10	133	105	140	150
P105	128	102	112	126
P12607	121	115	134	103
P114	121	138	114	116
P262219	117	112	112	121
Media	124	114,4	122,4	123,2
Desviación típica	6,403124237	14,18802312	13,52035502	17,25398505

Figura 3.26: Tabla con los tiempos de cargas de cada gráfica del ejercicio 3 correspondientes a cada paciente, la media y la desviación típica.

La Figura 3.28 muestra la tabla compuesta por las gráficas del ejercicio 3 y los cinco valores de tiempo de carga que se obtuvieron para los cinco pacientes. Es el origen del diagrama de caja y bigotes de la Figura 3.29.

Gráficas	Tiempos
Rango pulgar (ms)	133
Rango pulgar (ms)	128
Rango pulgar (ms)	121
Rango pulgar (ms)	121
Rango pulgar (ms)	117
Rango indice (ms)	105
Rango indice (ms)	102
Rango indice (ms)	115
Rango indice (ms)	138
Rango indice (ms)	112
Rango anular (ms)	140
Rango anular (ms)	112
Rango anular (ms)	134
Rango anular (ms)	114
Rango anular (ms)	112
Rango meñique (ms)	150
Rango meñique (ms)	126
Rango meñique (ms)	103
Rango meñique (ms)	116
Rango meñique (ms)	121

Figura 3.27: Tabla con todos los valores temporales de cada gráfica del ejercicio 3

La Figura 3.29 contiene el diagrama de caja y bigotes generado a partir de la tabla de la Figura 3.28. En dicho diagrama se puede observar como la petición que grafica el resultado de “Rango meñique” es el que presenta mayor variabilidad en sus datos, mientras que el resultado de “Rango pulgar” es el que presenta menor variabilidad.

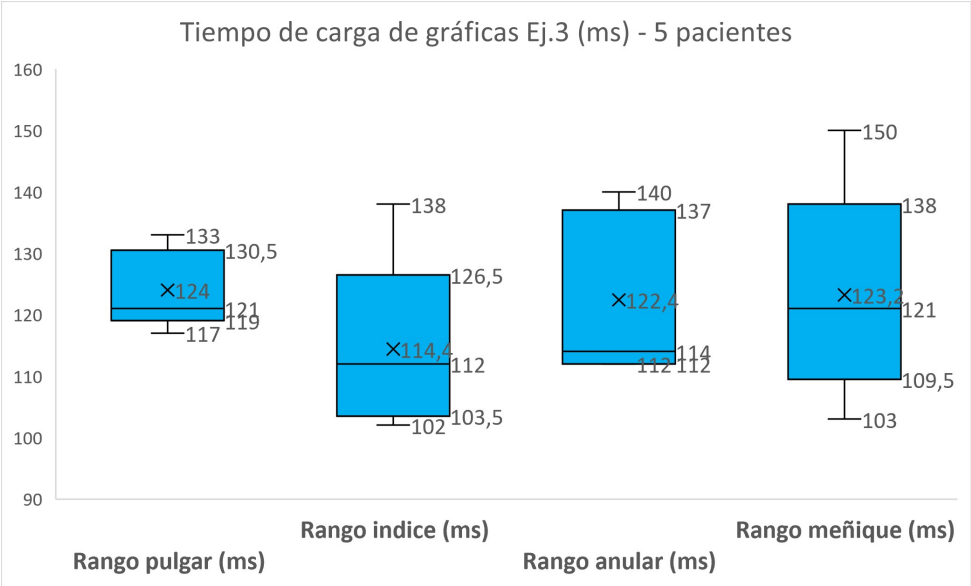


Figura 3.28: Diagrama de caja y bigotes de los tiempos de carga del ejercicio 3

Para el ejercicio 3 se obtuvieron los siguientes tiempos de carga medios y desviaciones típicas:

- Gráfica *Rango pulgar*:
  - Tiempo de carga medio = 124 ms
  - Desviación típica = 6,40 ms
- Gráfica *Rango índice*:
  - Tiempo de carga medio = 114,4 ms
  - Desviación típica = 14,19 ms
- Gráfica *Rango anular*:
  - Tiempo de carga medio = 122,4 ms
  - Desviación típica = 13,52 ms
- Gráfica *Rango meñique*:
  - Tiempo de carga medio = 123,2 ms
  - Desviación típica = 17,25 ms

Al comparar dichos resultados con el diagrama de caja y bigotes los valores de media coinciden, al igual que el tamaño de las cajas respecto a los valores de las desviación típica. La mayor desviación típica la posee la graficación del rango del dedo meñique, lo que concuerda con el gran tamaño de su caja correspondiente en el diagrama de la Figura 3.29. La graficación del rango del dedo pulgar tiene una desviación típica pequeña por lo que su valor de tiempo de carga es muy similar cada vez que se realiza la petición.

- **Tiempos de carga para gráficas del ejercicio 4:** la Figura 3.30 muestra la tabla de los tiempos de carga que se obtuvieron en las pruebas con los cinco pacientes. Además, contiene el valor de la media de dichos tiempos de carga y su desviación típica para el ejercicio 4

	Tiempos de carga de gráficas del ejercicio 4	
Paciente	Perímetro máximo (ms)	Rango pulgar (ms)
P10	203	202
P105	312	175
P12607	162	199
P114	180	164
P262219	175	183
Media	206,4	184,6
Desviación típica	60,86296082	16,04057356

Figura 3.29: Tabla con los tiempos de cargas de cada gráfica del ejercicio 4 correspondientes a cada paciente, la media y la desviación típica.

La Figura 3.31 muestra la tabla compuesta por las gráficas del ejercicio 4 y los cinco valores de tiempo de carga que se obtuvieron para los cinco pacientes. Es el origen del diagrama de caja y bigotes de la Figura 3.32.

Gráficas	Tiempos
Perímetro máximo (ms)	203
Perímetro máximo (ms)	312
Perímetro máximo (ms)	162
Perímetro máximo (ms)	180
Perímetro máximo (ms)	175
Rango pulgar (ms)	202
Rango pulgar (ms)	175
Rango pulgar (ms)	199
Rango pulgar (ms)	164
Rango pulgar (ms)	183

Figura 3.30: Tabla con todos los valores temporales de cada gráfica del ejercicio 4

La Figura 3.32 contiene el diagrama de caja y bigotes generado a partir de la tabla de la Figura 3.31. En dicho diagrama se puede observar como la petición que grafica el resultado de “Perímetro máximo” es el que presenta mayor variabilidad en sus datos, mientras que el resultado de “Rango pulgar” es el que presenta menor variabilidad.

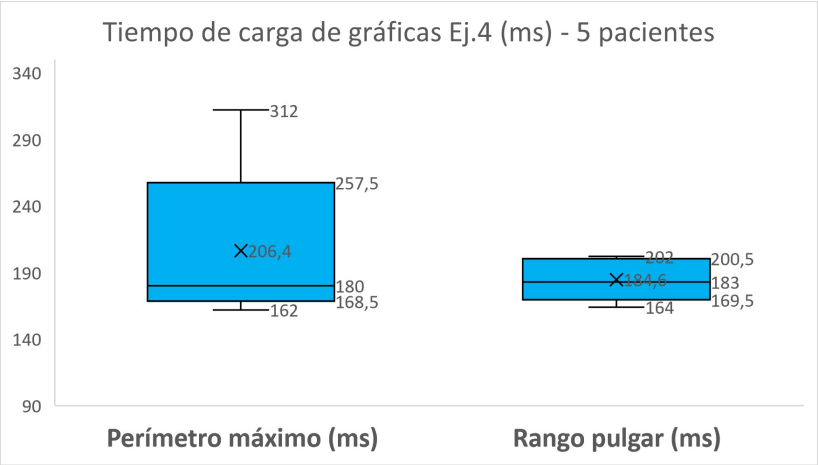


Figura 3.31: Diagrama de caja y bigotes de los tiempos de carga del ejercicio 4

Para el ejercicio 4 se obtuvieron los siguientes tiempos de carga medios y desviaciones típicas:

- Gráfica *Rango pulgar*:
  - Tiempo de carga medio = 206,4 ms
  - Desviación típica = 60,86 ms
- Gráfica *Rango índice*:
  - Tiempo de carga medio = 184,6 ms
  - Desviación típica = 16,04 ms

De los resultados comentados anteriormente se puede observar que las peticiones realizadas al *backend* para recuperar los datos de los ejercicios y graficarlos dependen de más factores que solo la configuración y rendimiento de la aplicación. Como se mencionó al inicio del apartado, el tiempo de carga puede variar dependiendo de la estabilidad, velocidad, latencia y demás características de la conexión a Internet a la que se tiene acceso.

Sin embargo, la finalidad de estos resultados es establecer una aproximación de los tiempos de carga medios para cada representación gráfica y evaluar el rendimiento de la aplicación en cuanto a sus respuestas temporales relacionadas a dichas gráficas. Y, tomando en cuenta los resultados, se puede asegurar que la aplicación posee una respuesta temporal media rápida que evita que el usuario sufra tiempos de espera indeseados.

Capítulo 4

Modulo de generación de contenido web automático para la divulgación de herramientas

En este capítulo se presenta un módulo de generación de contenido web orientado a la divulgación de proyectos tecnológicos y/o científicos relacionados con las enfermedades neurológicas, liderados por el equipo colaborativo formado por el Laboratorio de Robótica y Control de la ETSI de Telecomunicación de la UPM y Instituto para la Investigación Biomédica del Hospital Universitario La Paz-IdiPA

4.1. La divulgación científica a través de Internet y su importancia

Anteriormente los medios tradicionales como periódicos, revistas, televisión y radio eran los encargados de divulgar el conocimiento científico, pero la llegada del Internet y las Tecnologías de la Información y la Comunicación (TICs) supuso un aumento de plataformas digitales para dar a conocer esta clase de contenido y que el público objetivo tuviese más acceso [44].

En la actualidad, la red es el medio de comunicación más grande que existe pero en algunas ocasiones, en el campo de la investigación, no se le da la importancia que deberían tener a las herramientas que se pueden utilizar para explotar esta vía de difusión y divulgación de la información [45].

Siempre es necesario realizar publicaciones científicas que parten de un lenguaje profesional y especializado y que se orientan a una audencia de expertos o personas documentadas en dicha temática. Pero lo que se busca es ampliar el público al que va dirigida esta información e implementar diferentes formas de explicar este tipo de contenidos y de como se muestran al público [45].

La importancia de la divulgación científica va más alla de transmitir un mensaje. Se debe buscar que la sociedad sea participe de los conocimientos generados y que realmente se beneficie de los mismos [44]. Y para ello, lo mejor es utilizar las

herramientas que permitan captar la mayor cantidad de público general posible, las TICs y el Internet.

4.2. Descripción del módulo de generación de contenido web

El módulo de generación de contenido está formado por plantillas de código de diferentes lenguajes que permiten la automatización de la creación de una página web. A continuación se describen diferentes aspectos del módulo.

4.2.1. Estructura

Para la generación de cada página web, el módulo utiliza cuatro archivos diferentes que poseen una estrucutura de partida (por defecto) que puede ser editada según las características de la página que se esté creando. Dichos archivos son:

- Un archivo HTML.
- Un archivo CSS.
- Un archivo JavaScript.
- Un archivo JSON.

Cada página web cuenta con un *frontend* y un *backend*, conceptos descritos anteriormente. En este caso, al tratarse de una aplicación web estática, su desarrollo es más sencillo que para una aplicación web dinámica, como es el caso del capítulo 2.

Para el frontend se ha empleado HTML, CSS y JavaScript. Mientras que para el backend se ha utilizado JavaScript y JSON. A continuación se describen las tecnologías empleadas y sus funciones dentro del módulo.

4.2.1.1. Tecnologías empleadas

- **HTML:** se encarga de estructurar cada sección que conforma las páginas web creadas a partir de este módulo. Esta tecnología se ha definido anteriormente.
- **CSS:** otorga los estilos a la hora de presentar los elementos HTML en el frontend (colores, tamaños, fuentes, entre otros). Esta tecnología se ha definido anteriormente.
- **Bootstrap:** es una plataforma de código abierto que ofrece herrmientas basadas en lenguaje HTML y CSS para ser implementadas en aplicaciones web. Es común que se describa como un marco CSS [46]. En las secciones de las páginas web se utilizan diferentes elementos de Bootstrap para ofrecer una mejor experiencia al usuario visitante.
- **JavaScript:** se utiliza para crear elementos HTML que se integran en el *frontend* pero a la vez se encarga de recuperar la información del archivo JSON para

colocarla en la página web. Por ello cumple funciones tanto en el *backend* como en el *frontend*. Es el lenguaje de programación principal que permite que el módulo de creación de contenido sea automático. Este lenguaje se ha definido anteriormente

- **JSON:** el archivo JSON posee una estructura dividia por los diferentes campos que forman las secciones de cada página web. Por defecto es una plantilla que no posee valores en dichos campos para que al momento de crear el sitio web se rellenen con la información deseada. También se ha definido anteriormente.

4.2.2. Funcionamiento

El funcionamiento del módulo se divide en tres partes fundamentales:

- *Frontend* de cada página.
- *Backend* de cada página.
- Pagina general.

Cada parte se describe detalladamente a continuación:

4.2.2.1. Frontend

El *frontend* se compone de elementos HTML a los que se les aplica los estilos definidos en el archivo CSS (que incluye los estilos de Bootstrap y los personalizados). La plantilla posee elementos HTML fijados en el archivo HTML y otros que son creados e insertados en dicho archivo por el código desarrollado con JavaScript. Este código se compone de diferentes funciones, cada una correspondiente a la sección que debe crear en la página, y aporta creación de elementos y edición de algunos de los que ya están fijados en la plantilla.

Si se desea que alguna sección, de las que vienen por defecto en el módulo, no se muestre en el *frontend* basta con eliminar o comentar el código correspondiente a dicha sección tanto en el fichero HTML como en el fichero JavaScript. También se pueden agregar otras secciones en caso de ser necesario, pero teniendo en cuenta que cualquier sección nueva debe ser completamente configurada en todos los archivos que componen la página web.

Para lograr una comunicación exitosa con el *backend* se debe definir al fichero JavaScript en un elemento HTML de tipo *<script>* al final de la etiqueta *<body>* del archivo HTML.

4.2.2.2. Backend

El *backend* está formado por el archivo JSON que almacena toda la información de la página y la parte del código JavaScript que se encarga de recuperar dicha información para mostrarla en el *frontend*.

El archivo JavaScript, como se mencionó anteriormente, posee una serie de funciones que están destinadas a crear las secciones de la página web, incluyendo la barra de navegación. Cada función cuenta con el uso del método *fetch()*, que se ha explicado en capítulos previos. Dicho método le permite a cada función recuperar la información del archivo JSON y luego utilizarla para formar los elementos que posteriormente se incluyen en el *frontend*. Todas las veces que se utiliza este método en el fichero JavaScript se realizan peticiones HTTP de tipo "GET", ya que solo se desea recuperar la información.

Se logra comunicar con el frontend por su definición dentro del propio archivo HTML, como se comenta previamente en el aparto del *frontend*. Y su comunicación con el archivo JSON es a través de peticiones de tipo HTTP realizadas con el método *fetch()* de JavaScript.

4.2.2.3. Pagina general

La pagina web general es la que está orientada a ser la entrada a todos los proyectos cuya página se ha creado a partir de este módulo y que pertenecen al equipo colaborativo de ROBOLABO y el HULP.

Se compone de un *frontend* y un *backend* que coinciden con los descritos anteriormente para las páginas de las herramientas en cuanto a los archivos necesarios para su funcionamiento, las tecnologías empleadas y los estilos definidos en el archivo CSS respectivo. Pero son diferentes en la configuración de la estructura del *frontend* propuesta en el HTML, de las funciones del fichero JavaScript y de la división del archivo JSON.

Es una página que está destinada a ser la presentación del grupo colaborativo de ROBOLABO y el Instituto para la Investigación Biomédica del Hospital Universitario La Paz-IdiPAZ, además de contener los enlaces que redirigen a los usuarios a lás páginas web de los diferentes proyectos que se exponen y la información de los integrantes del equipo.

El *frontend* de esta página posee cuatro secciones solamente: *Home* o inicio, descripción del grupo colaborativo, la presentación de los proyectos y los integrantes del equipo. Además, cuenta con un pie de página que coincide con el de las páginas de los proyectos, en donde se presentan los logos de las instituciones que han participado en todos los proyectos y, a su vez, son un enlace a sus respectivas páginas web.

El *backend* posee funciones y una estructura del archivo JSON diferentes debido a que las secciones son diferentes para esta página web. Sin embargo, el funcionamiento de dichas funciones se basa en el mismo método *fetch()* de JavaScript para recuperar infromación y también crean elementos y los editan para mostrarlos en el *frontend*.

4.3. Proyectos actuales

Actualmente, el grupo colaborativo de ROBOLABO y el Instituto para la Investigación Biomédica del Hospital Universitario La Paz-IdiPAZ ha llevado a cabo diferentes proyectos para el desarrollo de herramientas relacionadas con diferentes problemáticas referentes a enfermedades neurológicas. Y con este módulo de generación de contenido web automático se han generado una serie de páginas web, para dichas herramientas, que se han subido al servidor de ROBOLABO.

Capítulo 5

Conclusiones

5.1. Conclusiones

En este Trabajo de Fin de Grado se han cumplido los objetivos establecidos en el *Capítulo 1* para diseñar e implementar tanto la aplicación de visualización de datos como el módulo de generación de contenido web automático.

En cuanto a la aplicación web de visualización de datos, se logró desarrollar un *backend* totalmente funcional que se comunica con una base de datos, alojada en el servidor web del Laboratorio de Robótica y Control de la ETSIT-UPM (ROBOLABO), y es capaz de recuperar datos de dicha base de datos para procesarlos mediante una serie de operaciones matemáticas y posteriormente comunicarse con el *frontend* para mostrar la información. Para completar el desarrollo del *backend* se empleó Python. Además, se desarrolló un *frontend*, utilizando React, para complementar el *backend* de la aplicación de visualización de datos y garantizar una interfaz de usuario intuitiva y fácil de utilizar. Esta permite mostrar toda la información necesaria en diferentes vistas que, a su vez, cuentan con distintas gráficas que se muestran rápidamente en el *frontend* debido a los cortos tiempos de respuesta de la aplicación. El acceso a esta aplicación es, principalmente, mediante la página web del dispositivo NDT que se menciona en capítulos previos.

Se han cumplido los objetivos referentes a la comunicación entre ambas partes de la aplicación web y del *backend* con la base de datos. Esto debido al uso de tecnologías como JavaScript o Flask que son los encargados de la comunicación *frontend-backend* y de la comunicación *backend-base de datos* respectivamente. Esto asegura el correcto funcionamiento de la aplicación y, por lo tanto, el cumplimiento de su objetivo principal que es la visualización de datos de pacientes.

Por otro lado, también se han cumplido los objetivos referentes al módulo de generación de contenido web automático. Se utilizó HTML, CSS y JSON para crear tres de las cuatro plantillas web que forman parte de este módulo. Estas son para estructurar y estilizar la página, además de la plantilla JSON que alberga la información que se desea publicar. La plantilla restante es de código JavaScript, encargada de otorgar automatización a la generación de las páginas web. Las páginas

web que se han creado a partir de este módulo se han publicado en el servidor web de ROBOLABO con la finalidad de cumplir el objetivo principal de esta iniciativa, la divulgación de las herramientas desarrolladas en los proyectos del equipo colaborativo de ROBOLABO con el Instituto para la Investigación Biomédica del Hospital Universitario La Paz-IdiPAZ a través de la red que, como se ha mencionado en el *Capítulo 4*, es el medio de comunicación más grande e importante de la actualidad.

Durante el desarrollo del módulo de generación de contenido web automático no se presentaron limitantes relevantes. Sin embargo, durante el desarrollo de la aplicación de visualización de datos se presentaron dificultades a la hora de configurar tanto el *frontend* como el *backend* y la comunicación entre ellos. Además, el desarrollo del *backend* se inició utilizando el lenguaje de programación PHP, pero debido al procesado de datos se tuvo que cambiar dicho lenguaje por Python para poder completar el desarrollo del *backend*. Dichos problemas se han superado y se ha logrado configurar correctamente dichos aspectos para obtener una aplicación completamente funcional.

5.2. Líneas futuras

A continuación se comentan las líneas futuras que pueden ayudar a realizar un desarrollo más profundo tanto para la aplicación de visualización de datos como para el módulo de generación de contenido web automático:

- Para la aplicación web de visualización de datos:
  - Implementar medidas de seguridad para proteger la aplicación y los datos que se utilizan en ella. Esto incluye un portal de acceso que funcione con conexión a la base de datos.
  - Implementar un sistema de búsqueda para la localización de pacientes en la vista inicial (*Home*) de la aplicación, que contiene el listado de los pacientes.
  - Añadir una vista nueva, como la de un ejercicio, pero que muestre los datos personales y antecedentes clínicos del paciente que sean relevantes.
  - Agregar un botón de "volver a atrás" para navegar entre las vistas de la aplicación de una manera más cómoda.
- Para el módulo de generación de contenido web automático:
  - Implementar una opción de cambio de idioma entre el inglés y el castellano, para que las páginas web creadas a partir del módulo estén disponibles en ambos idiomas.
  - Profundizar en los elementos y estilos que se implementan en la página web para ofrecer sitios web modernos que capten, aún más, la atención del usuario.

Referencias

[1] Luján S. *Programación de aplicaciones web: Historia, Principios básicos y Clientes web.* Editorial Club Universitario, C/. Cottolengo, 25 - San Vicente (Alicante), 2002.

[2] Franzolini D. Los 7 tipos de aplicaciones web que existen (y ejemplos). `https://blog.hubspot.es/website/tipos-aplicaciones-web`. Consultado: 16/05/2023.

[3] Millán J. *Desarrollo de sitios web Dinámicos*, pages 91–97. Asociación de Autores Científico-Técnicos y Académicos., 2008. Consultado: 17/05/2023.

[4] Torrecilla A. Gestión de datos de la investigación. Master's thesis, Universidad Politécnica de Valencia, 2013. `https://riunet.upv.es/bitstream/handle/10251/36053/Tesina%20final%20de%20estudios%20M%C3%A1ster%20Oficial%20CALSI.pdf?sequence=1`.

[5] Alonso-Arévalo J. La gestión de datos de investigación en el horizonte de las bibliotecas universitarias y de investigación. *Cuadernos de Documentación Multimedia*, 30:75–88, 2018. `https://doi.org/10.5209/CDMU.62806`.

[6] Bilbao R, Belar O, Ezkerra I, Bidaurrazaga J, Amiano P, González N, Sola C, Arrúe M, Epelde G, Stopar L, Fuart F, and Costa J. Desarrollo de un proyecto de big data en salud pública: proyecto midas. `https://www.fundacionmercksalud.com/wp-content/uploads/2020/03/2.4.-PROYECTO-BIG-DATA-EN-SALUD-PUBLICA.-FVIIS-Bionbanco-Vasco.pdf`. Consultado: 2023-06-19.

[7] Fundación Gaspar Casal. *Big Data, Datos de Vida Real y evaluación de nuevos medicamentos.* Libroacadémico, S.L., C/ General Díaz Porlier 78, 8A, 28006, Madrid, 1 edition, 2021.

[8] López-Blanco R, Sorrentino Rodriguez A, Cubo E, Gabilondo Í, Ezpeleta D, Labrador-Espinosa M.A, Sánchez-Ferro A, Tejero C, and Matarazzo M. Impacto de las nuevas tecnologías en la neurología en españa. revisión del comité ad-hoc de nuevas tecnologías de la sociedad española de neurología. *Neurología - Sociedad Española de Neurología*, 2020. doi: `https://doi.org/10.1016/j.nrl.2020.10.015`.

[9] Hatem SM, Saussez G, della Faille M, Prist V, Zhang X, Dispa D, and Bleyenheuft Y. Rehabilitation of motor function after stroke: A multiple systematic review focused on techniques to stimulate upper extremity recovery. *Frontiers in Human Neuroscience*, 2016. doi: `https://doi.org/10.3389/fnhum.2016.00442`.

[10] Pérez A. 29 de octubre: Día mundial del ictus. *Sociedad Española de Neurología - Departamento de Prensa*, 2017. https://www.sen.es/saladeprensa/pdf/Link223.pdf.

[11] Ustrell-Roig X and Serena-Leal J. Ictus. diagnóstico y tratamiento de las enfermedades cerebrovasculares. *Revista Española de Cardiología*, 2007. https://www.revespcardiol.org/es-ictus-diagnostico-tratamiento-enfermedades-cerebrovasculares\-articulo-13108281.

[12] Choi MJ, Kim H, Nah HW, and Kang DW. Digital therapeutics: Emerging new therapy for neurologic deficits after stroke. *Journal of Stroke*, 2019. doi: https://doi.org/10.5853/jos.2019.01963.

[13] Gutiérrez Zúñiga R, Alonso de Leciñana M, Díez A, Torres Iglesias G, Pascual A, Higashi A, Rodríguez Pardo J, Hernández Herrero D, Fuentes B, and Díez Tejedor E. A new software for quantifying motor deficit after stroke: A case-control feasibility pilot study. *Frontiers in neurology*, 2021. doi: https://doi.org/10.3389/fneur.2021.603619.

[14] Fernández Y and Díaz Y. Patrón modelo-vista-controlador. *Telemática*, 11:47–57, 2012. https://revistatelematica.cujae.edu.cu/index.php/tele/article/view/15.

[15] Flórez H. *Programación Orientada a Objetos usando Java*. Ecoe Ediciones, 2012.

[16] International Business Machines Corporation (IBM). ¿qué es mongodb? https://www.ibm.com/es-es/topics/mongodb.

[17] MongoDB. Compass. la gui para mongodb. https://www.mongodb.com/es/products/compass#:~:text=La%20GUI%20para%20MongoDB.,desarrollar%20canales%20y%20mucho%20m%C3%A1s.&text=Trabaje%20f%C3%A1cilmente%20con%20sus%20datos,desarrollado%20por%20y%20para%20MongoDB.

[18] mdn web docs. Html: Lenguaje de etiquetas de hipertexto. https://developer.mozilla.org/es/docs/Web/HTML.

[19] mdn web docs. Css. https://developer.mozilla.org/es/docs/Web/CSS.

[20] mdn web docs. Javascript. https://developer.mozilla.org/es/docs/Web/JavaScript.

[21] Redacción KeepCoding. Lenguajes de programación interpretados vs compilados. https://keepcoding.io/blog/lenguajes-programacion-interpretados-compilados/.

[22] mdn web docs. Getting started with react. https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/React_getting_started.

[23] w3schools. Chart.js. https://www.w3schools.com/ai/ai_chartjs.asp.

[24] mdn web docs. Python. https://developer.mozilla.org/es/docs/Glossary/Python.

[25] MongoDB. *PyMongo*. https://www.mongodb.com/docs/drivers/pymongo/.

[26] Redacción KeepCoding. ¿qué es flask? https://keepcoding.io/blog/que-es-flask/.

[27] Marques H. ¿qué es un desarrollador frontend y qué hace? https://marquesfernandes.com/es/tecnologia-es/que-es-un-frontend-desarrollador-y-que-hace/, Agosto 2020.

[28] Ferry A. Introducción al frontend y backend. Publicado en: https://openaccess.uoc.edu/bitstream/10609/141486/1/Tecnologias%20y%20herramientas%20para%20el%20desarrollo%20web_Modulo1_Introduccion%20al%20frontend%20y%20backend.pdf.

[29] López D, Casado-Fernández L, Fernández F, Fuentes B, Larraga-García B, Rodríguez-Pardo J, Hernández D, Alonso E, Díez-Tejedor, Gutiérrez A, and Alonso M. Neurodata tracker: Software for computational assessment of hand motor skills based on optical motion capture in a virtual environment. *DIGITAL HEALTH*, 9, 2023. doi: https://doi.org/10.1177/20552076231174786.

[30] Coppola M. ¿qué es una api? definición, tipos y ejemplos. https://blog.hubspot.es/website/que-como-usar-api. Consultado: 22/06/2023.

[31] International Business Machines Corporation (IBM). Formato json (javascript object notation. https://www.ibm.com/docs/es/baw/20.x?topic=formats-javascript-object-notation-json-format.

[32] mdn web docs. Fetch api. https://developer.mozilla.org/es/docs/Web/API/Fetch_API.

[33] mdn web docs. Uso de fetch. https://developer.mozilla.org/es/docs/Web/API/Fetch_API/Using_Fetch.

[34] mdn web docs. Mensajes http. https://developer.mozilla.org/es/docs/Web/HTTP/Messages.

[35] mdn web docs. Http request methods. https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods.

[36] Tierz J. Reconocimiento e interpretación de gestos con dispositivo leap, 2013.

[37] Álvaro (Fisiotherapymadrid). Lesiones en la articulación de la muñeca. https://www.fisiotherapymadrid.com/post/lesiones-mu%C3%B1eca, 2022.

[38] Grzejszczak T and Niezabitowski M. Applications of hand feature points detection and localization algorithms, 2016. https://www.matec-conferences.org/articles/matecconf/pdf/2016/19/matecconf_iccae2016_02009.pdf.

[39] Lefan Wang, Turgut Meydan, and Paul Ieuan Williams. A two-axis goniometric sensor for tracking finger motion. *Sensors*, 17(4), 2017. doi: https://doi.org/10.3390/s17040770.

[40] Fakhfakh S and Ben Y. Gesture recognition system for isolated word sign language based on key-point trajectory matrix. *Computación y Sistemas*, 22(4), 2018. doi: https://doi.org/10.13053/cys-22-4-3046.

[41] mdn web docs. Preflight petición. https://developer.mozilla.org/es/docs/Glossary/Preflight_request.

[42] cingles. ¿qué es la velocidad de subida y bajada de la fibra óptica? https://cinglescomunicacions.com/es/velocidad-subida-bajada-fibra-optica/#:~:text=La%20velocidad%20de%20bajada%20es,desde%20nuestro%20ordenador%20o%20m%C3%B3vil.

[43] Movistar. ¿qué es la latencia y cómo se puede mejorar? https://ww2.movistar.cl/blog/post/que-es-la-latencia/, 2022.

[44] Múnera M and Marín B. La divulgación científica en la web, un panorama latinoamericano. *Comunicación*, 31:35–41, 2014. https://dialnet.unirioja.es/servlet/articulo?codigo=5470103.

[45] Gobierno del Principado de Asturias. *Difusión y divulgacióncientífica en Internet*. Gobierno del Principado de Asturias, 2011.

[46] Londoño P. Qué es bootstrap, para qué sirve y cómo funciona. https://blog.hubspot.es/website/que-es-bootstrap. Consultado: 25/06/2023.

[47] Agramonte A. Implicaciones éticas del uso de aplicaciones informáticas en la gestión de cuidados de enfermería. *Revista Cubana de Enfermería*, 29:199–209, 2013. http://scielo.sld.cu/pdf/enf/v29n3/enf06313.pdf.

[48] talent.com. Salario en españa 2023. https://es.talent.com/es/salary. Consultado: 26/06/2023.

[49] Hands on React. Build & deploy. https://handsonreact.com/docs/build-deploy#:~:text=npm%20run%20build%20creates%20a,enables%20long%20term%20caching%20techniques. Consultado: 27/06/2023.

[50] Andrés R. Qué es .htaccess, para qué sirve y códigos fundamentales. https://computerhoy.com/noticias/internet/que-es-htaccess-que-sirve-codigos-fundamentales-76211. Consultado: 27/06/2023.

[51] mdn web docs. Intercambio de recursos de origen cruzado (cors). https://developer.mozilla.org/es/docs/Web/HTTP/CORS.

[52] J.D Gauchat. *El gran libro de HTML5, CSS3 y JavaScript*. Marcombo, Gran Via de les Corts Catalanes, 594, 08007, Barcelona., 3 edition, 2017.

[53] Bootstrap. Bootstrap. https://getbootstrap.com/. Consultado: 2023-04-03.

[54] J. Angulo e I. Botero. Stack overflow. https://es.stackoverflow.com/questions/25088/cu%C3%A1l-es-el-mejor-lugar-para-colocar-los-tag-scripts-src-en-html, 2017. Consultado: 2023-04-03.

[55] MDN. Mdn web docs. https://developer.mozilla.org/es/. Consultado: 2023-04-06.

[56] Understanding ssh. https://winscp.net/eng/docs/ssh, s.f. Consultado: 2023-06-01.

Apéndice A

Aspectos éticos, económicos, sociales y ambientales

A.1. Introducción

El desarrollo de una aplicación de visualización de datos de pacientes con déficit motor supone una herramienta con un gran potencial dentro del área de la neurología y con un impacto significativo en diferentes aspectos, ya que permite mejorar los diagnósticos, las monitorizaciones y los seguimientos que se realizan a dichos pacientes. Esto causa grandes impactos en lo social y económico que se desarrollan a continuación, además de los respectivos aportes en cuanto a los aspectos éticos y ambientales.

Además, también se expone el desarrollo de un módulo automático de creación de páginas web para la divulgación de herramientas y proyectos del grupo colaborativo mencionado anteriormente, con el fin de generar, principalmente, un impacto social en el público alcanzado por las páginas web que componen este módulo. También se comentan los respectivos aspectos económicos, éticos y ambientales.

A.2. Descripción de impactos relevantes relacionados con el proyecto

- **Aspecto ético:** aunque no es el aspecto en el que el TFG tiene mayor impacto, se debe tomar en cuenta ya que se presentan temas importantes como la privacidad y seguridad de los datos de los pacientes que se procesan y grafican en la aplicación.

  En cuanto a la ética informática relacionada al uso de aplicaciones dentro de un entorno sanitario, hay diferentes principios que deben cumplirse y entre ellos se encuentran: la privacidad y disposición de la información, seguridad, confidencialidad, entre otras [47]. Para cumplir con ello, la aplicación cuenta con una lista de pacientes identificados por un *id* que se les asocia en la base de datos y con un portal de acceso que, a pesar de ser de baja seguridad, es un punto de partida para un módulo de seguridad completo en versiones futuras.

En cuanto al módulo de contenido web, este carece de un gran impacto ético mas allá de la responsibilidad relacionada a la veracidad de la información que se divulga y la forma en que se diseminan las herramientas.

- **Aspecto económico:** tanto la aplicación web como el módulo de generación de contenido cuentan con un impacto económico, en su mayoría, positivo. Ambas iniciativas se han desarrollado con el uso de tecnologías y lenguajes de programación de código abierto que evitan el uso de tecnologías que requieran una inversión económica o algún tipo de membresía, lo que las hace muy accesibles. Aunque el tipo de aplicaciones que se han desarrollado en este TFG conllevan un almacenamiento y mantenimiento que a largo plazo puede ser muy costoso

- **Aspecto social:** tanto la aplicación web como el módulo de generación de contenido cuentan con un impacto social significativo, directo y positivo. Por un lado, la aplicación tiene un impacto tanto en profesionales de la salud como en los pacientes, ya que los profesionales se benefecian de una herramienta que los ayuda a ofrecer un mejor y más preciso servicio sanitario, concretamente en el área de la neurología. Y por otro, el contenido web generado a partir del módulo automático busca captar la mayor cantidad de público general para dar a conocer las herramientas.

- **Aspecto ambiental:** en cuanto al medio ambiente, no se puede establecer un impacto muy concreto por parte de las dos ideas desarrolladas en este TFG, ya que al ser aplicaciones web, se pueden ejecutar en cualquier ordenador (en el caso de las páginas web) o en los ordenadores de una red interna de un hospital (en el caso de la apliacción de visualización de datos). Ambas contribuyen al fenómeno de la digitalización de la información, lo que ayuda a utilizar en menor medida algunos medios tradicionales para almacenar información y/o divulgarla, como el papel, el periódico o revistas, lo cual genera un impacto positivo sobre el medio ambiente, pero no a gran escala.

A.3. Análisis detallado de alguno de los principales impactos

El principal impacto, tanto de la aplicación de visualización de datos como del módulo de generación de contenido web, es sobre el aspecto social y el aspecto económico.

En el caso de la aplicación para la visualización de datos, el impacto social generado se divide en los beneficios que se obtienen en el sistema de salud y los beneficios que se obtienen en la salud de las personas que sufren déficit motor como consecuencia de algún accidente o patología. De esta manera, se busca que los profesionales de la salud del área de neurología cuenten con una herramienta que complemente su labor y los ayude a ofrecer una medicina personalizada más precisa a cada paciente, lo cual

engloba tratamientos, procesos de rehabilitación y la monitorización de la evolución de cada paciente. Además, es un recurso que ofrece fácil acceso a los datos obtenidos en las pruebas clínicas con el dispositivo NeuroData Tracker, mediante una interfaz de usuario muy intuitiva que tiene mucho potencial de mejora.

Por otro lado, el módulo de generacíon de contenido web automático tiene el objetivo de llevar la información de los proyectos tecnológicos y/o científicos, desarrollados por el grupo colaborativo nombrado anteriormente, a la mayor cantidad de público posible para que se documenten acerca de ellos y se interesen por involucrarse en ellos de alguna forma. Además de ser un método innovador de dar a conocer información científica de una forma muy diferente a los métoos tradicionales. Como se comentó en el capítulo 4, la divulgación va más alla de transmitir un mensaje, se trata sobre que la sociedad pueda captar la información y que les sea de utilidad en sus vidas.

En cuanto al impacto económico, como se comentó previamente, a corto plazo es positivo ya que, más alla de algunos recursos informáticos, no se requiere de materia prima muy costosa para su desarrollo. Sin embargo, las aplicaciones web conllevan un almacenamiento y mantenimiento al finalizar su desarrollo, lo cual puede ser costoso a largo plazo, dependiendo de las características de cada aplicación. En el caso de las páginas web, al ser aplicaciones estáticas, no requieren un mantenimiento costoso debido a que son aplicaciones desarrolladas con el objetivo de ser lo mas simples posibles. Pero en el caso de aplicaciones dinamicas, como es la aplicación para la visualización de datos, el almacenamiento y mantenimiento puede requerir una gran inversión económica ya que poseen una estructura más compleja, se componen de más elementos que las aplicaciones estáticas, lo común es que estén vinculadas a bases de datos que también requieren un cuidado para garantizar un correcto funcionamiento y suelen derivar en sistemas que requieren como característica principal la escalabilidad, por lo que la inversión de recursos económicos e informáticos es mayor.

A.4. Conclusiones

El desarrollo e implementación de una aplicación de visualización de datos para evaluar paciente con déficit motor y un módulo de generación de contenido, que cumplen con los objetivos propuestos, tienen un impacto general positivo. Siendo el aspecto social y el aspecto ecnómico los dos más involucrados. También se tienen un impacto positivo en cuanto al medio ambiente. El impacto sobre el aspecto ético será mayor a largo plazo.

Apéndice B

Presupuesto económico

El desarrollo de este Trabajo Fin de Grado se ha logrado debido a la colaboración del departamento de Tecnología Fotónica y Bioingeniería y del Servicio de Neurología del Hospital Universitario La Paz-IdiPAZ. A continuación se presenta la estimación del presupuesto económico en función del personal y los recursos utilizados en un lapso de 5 meses (duración del TFG):

- **Personal**: para el personal involucrado en el desarrollo de este TFG,se han tomado en cuenta al tutor académico, a la tutora profesional y al alumno (estudiante de ingeniería). La información del coste horario es una aproximación basada en información consultada en Internet [48]. (ver Tabla B.1).

	Coste horario (€)	**Horas**	**Total (€)**
Tutor académico (ingeniero)	19,5	30	585
Tutor profesional (médico)	24	20	480
Estudiante de ingeniería	12	360	4320
TOTAL			**5385**

Tabla B.1: Costes de personal.

- **Costes de recursos materiales**: el desarrollo del TFG se ha realizado con un único recurso tecnológico, un ordenador ASUS TUF Gaming FX505GM que posee un procesador Intel Core i7, un total de 16 GB de memoria RAM, almacenamiento interno de 237 GB y un discoduro adicional incorporado de 1TB (ver Tabla B.2). En la siguiente tabla (Tabla B.2) se establece la estimación del coste que ha representado el uso de este dispositivo como herramienta única y principal para el desarrollo del Trabajo Fin de Grado.

	Tiempo de vida (años)	Uds.	Coste (€)	Amortización (€/mes)	Uso (meses)	Total (€)
Ordenador ASUS	5	1	900	15	5	75
TOTAL						**75**

Tabla B.2: Costes de recursos materiales.

- **Formación complementaria**: para el desarrollo de este Trabajo Fin de Grado se realizó una inversión económica en dos cursos *en linea* sobre desarrollo web y un curso de ReactJS. El coste de esta inversión se presenta en la Tabla B.3:

	Coste
Curso "The web developer bootcamp 2023"	84,99€
Curso "React - The Complete Guide 2023"	84,99€
Curso "Desarrollo Web Completo 2.0"	84,99€
Total	254,97€

Tabla B.3: Costes de formación complementaria.

El coste total de las iniciativas desarrolladas en este Trabajo Fin de Grado se presenta la Tabla B.4, que contiene los costes que se han calculado previamente y se añade el *Impuesto sobre el Valor Añadido*, que actualmente posee un valor de 21 %.

	Coste
Costes de personal	5385€
Costes de material	75€
Costes de formación complementaria	254,97€
Subtotal	5714,97€
IVA	1200,14€
Total	6915,11€

Tabla B.4: Costes totales.

El coste total final del diseño e implementación de aplicaciones web para la divulgación y visualización de herramientas de evaluación de pacientes con déficit motor es de **6915,11€**.

Apéndice C

Manual de desarrollador - Aplicación de visualización de datos

Este manual de desarrollador va dirigido a profundizar la estructura y funcionamiento del código que compone tanto el *frontend* como el *backend* de la aplicación web de visualización de datos, con el fin de editar la interfaz de usuario, agregar funcionalidades o mejorar las que ya posee.

Con la finalidad de contextualizar al lector, se debe aclarar que la aplicacion web de visualización de datos posee una interfaz de usuario que utiliza como idioma principal el español, pero sus recursos asociados al *frontend* y *backend* poseen como idioma principal el inglés. Por ello, la información que se muestra en los archivos en ambos se encuentra en este mismo idioma, así como los nombres de ficheros y carpetas.

Como aclaratoria, se menciona que este manual no presenta el código de los diferentes archivos que se comentan ya que resultaría en una longitud excesiva del manual. Por lo que se recomienda descargar el archivo comprimido *reactJS.zip* en el siguiente enlace: `http://www.robolabo.etsit.upm.es/NDT/templates/`, descomprimirlo en local y abrir los archivos al momento de leer este manual, así se entenderán mucho mejor los temas que se explican en los siguientes apartados.

Estructura de archivos de la aplicación y su contenido:

En la Figura C.1 se presenta la estructura de las carpetas que componen tanto el *frontend* como el *backend*. Posteriormente se comentan sus contenidos y se explican las funciones de cada uno de los ficheros.

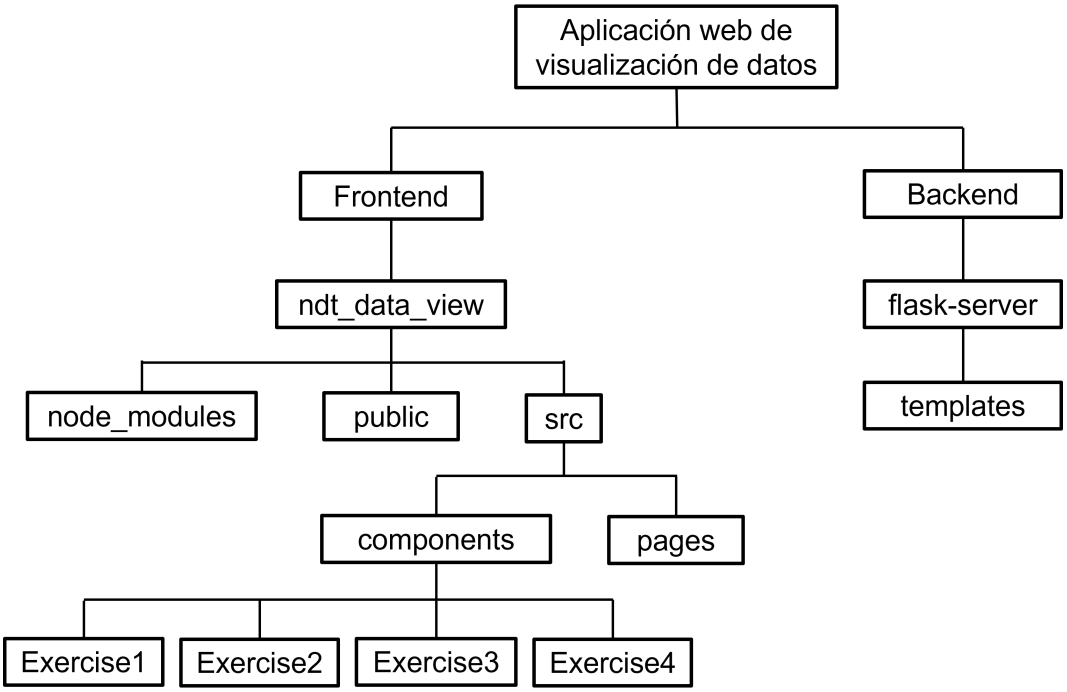


Figura C.1: Estructura de carpetas en desarrollo local del *frontend* y *backend*

Frontend:

Se debe mencionar que la estructura de los archivos del *frontend* de la aplicación al montarla en el servidor no es igual a su estructura a la hora del desarrollo en local. Para poder cargar la aplicación en el servidor web de ROBOLABO primero se debe ejecutar un comando en consola que genera una carpeta llamada *build* donde se almacena una versión optimizada de los archivos JavaScript y CSS que se han creado para el *frontend*. Se ha preparado una versión de producción de la aplicación creada con React [49]. La estructura que se explica en este apartado se refiere a la que se tiene al momento del desarrollo en local.

Primero se debe establecer los contenidos de las carpetas que componen el *frontend*:

- ***node_ modules***: esta carpeta contiene las dependencias descargadas mediante los comandos npm.
- ***public***: contiene archivos para el funcionamiento del *frontend*. El más importante es el fichero *index.html* ya que es donde se cargan los componentes del *frontend*.
- ***src***: contiene archivos para el funcionamiento del *frontend* y dos carpetas, *components* y *pages*. Uno de los archivos más relevantes que contiene es el fichero *App.js* que es el responsable de conectar los componentes con el fichero *index.html*. Además, posee los archivos de estilos que se aplican a cada uno de los archivos mencionados. Por último, se debe destacar el archivo *.htaccess* que se encarga de dirigir todas las peticiones al fichero *index.html* [50].
- ***components***: contiene los componentes que forman la interfaz de usuario de la

aplicación y sus archivos de estilos. Además, contiene un fichero CSS denominado *compBS.css* que es el que permite utilizar los elementos de *react-bootstrap*, una de las dependencias descargas que permite implementar componentes de Bootstrap en la aplicación.

- ***pages***: contiene otros componentes que forman las vistas de la aplicación a partir de los componentes de la carpeta anterior y sus respectivos archivos de estilos.
- **Carpetas *Exercise#***: están contenidas en la carpeta *components* y poseen los componentes que crean las gráficas de cada ejercicio con sus respectivos archivos CSS de estilos. El caracter "#" del nombre se refiere a los valores que toma dicho caracter, desde el 1 al 4, para cada ejercicio.

La ventaja de utilizar React es que se puede utilizar JavaSript y HTML en un mismo archivo de extension *.js*. Además de que acepta la inclusión de archivos de estilos CSS en dichos ficheros. Por lo tanto, aunque no se presente el código de los ficheros del *frontend*, se debe tener en cuenta que su contenido se compone de JavaScript y los resultados que devuelven son elementos HTML que forman los componentes.

Una vez establecida la estructura de las carpetas y su contenido, además de los lenguajes que componen los ficheros *.js*, a continuación se comentan los archivos más importantes que componen el *frontend*:

- ***App.js***: este archivo utiliza *createHashRouter* u *RouterProvider* para crear las rutas del *frontend* y que cada una lleve asiciada el componente React que debe renderizar. En este caso hay seis rutas: la ruta inicial de la vista de inicio de sesión, la vista *Home* donde se tiene la lista de pacientes y luego las cuatro rutas de los cuatro ejercicios. Además, es el archivo que se conecta con el archivo *index.html* para cargar todo el *frontend* en dicho fichero.
- ***Home.js***: este archivo contiene una función que solo devuelve un elemento HTML que contiene el componente de la lista de pacientes acompañada del título principal de la aplicación.
- ***LogIn.js***: contiene una función que devuelve el formulario de inicio de sesión. Se conecta con el *backend* a través del método *fetch()* de JavaScript, con una petición tipo GET, para recuperar la información referente al usuario y contraseña. Posee una función que compara los valores introducidos con la información recuperada, y si cumple con ser los mismos valores, redirige al usuario a la vista *Home* al presionar el boton de inicio de sesión.
- ***PatientsList.js***: es el componente que genera la lista de pacientes de la vista *Home*. Para recuperar la información de los pacientes utiliza el método *fetch()* de JavaScript con una petición tipo GET, específicamente utiliza los campos *username* y *id* de la base de datos. La lista que devuelve esta formada por un elemento de *react-bootstrap* denominado .Acordeón", que es el responsable de que al clicar en los *id* de cada paciente se muestren los botones de cada ejercicio en un desplegable. Además, utiliza un bucle sobre la lista de usuarios (*username)* para poder colocar a todos los pacientes de la base de datos. En el código JavaScript las variables de los usuarios y los *id* son: *datos.usernames* y *datos.ids*.

- **Archivos de las carpetas *Exercise#***: se encuentran los ficheros JavaScript que grafican los resultados de ambas manos, de la mano izquierda y la mano derecha respectivamente. Para ello, cada uno cuenta con una función que utiliza el método *fetch* de JavaScript para recuperar los datos del ejercicio correspondiente, a través de una petición de tipo POST para que el backend obtenga el usuario del paciente que se desea. La respuesta obtenida de dicha petición se convierte a formato JSON y, posteriormente, con el uso de Chart.js (librería de gráficos) se crean las gráficas con los resultados del ejercicio. Poseen las configuraciones necesarias para que los colores utilizados sean los correctos dependiendo de los valores obtenidos en el ejercicio. Además, se encuentran los archivos CSS para los estilos que se aplican en los elementos que componen la vista de las gráficas. El caracter "#" del nombre se refiere a los valores que toma dicho caracter, desde el 1 al 4, para cada ejercicio. Los archivos de estas carpetas son:
  - **Carpeta *Exercise1***:
    - ***Graph_ ex1.css***: estilos de las gráficas de los archivos *GraphL_ ex1.js* y *GraphR_ ex1.js*
    - ***GraphBoth_ ex1.css***: estilos de las gráficas del archivo *GraphBoth_ ex1.js.*
    - ***GraphBoth_ ex1.js***: grafica los resultados obtenidos, para ambas manos, de las variables: **vector normal a la palma de la mano** y **angulo respecto a la horizontal**. En el código JavaScript dichas variables son: lpnvZ, rpnvZ (vectores de mano izquierda y derecha respectivamente), *datos.ampHorL* y *datos.ampHorR* (ángulos de mano izquierda y derecha respectivamente).
    - ***GraphL_ ex1.js***: grafica los resultados obtenidos, para la mano izquierda, de las variables: **vector normal a la palma de la mano** y **angulo respecto a la horizontal**. En el código JavaScript dichas variables son: *datos.lpnvZ* (vector mano izquierda) y *datos.ampHorL* (ángulo de mano izquierda).
    - ***GraphR_ ex1.js***: grafica los resultados obtenidos, para la mano derecha, de las variables: **vector normal a la palma de la mano** y **angulo respecto a la horizontal**. En el código JavaScript dichas variables son: *datos.rpnvZ* (vector mano derecha) y *datos.ampHorR* (ángulo de mano derecha).
  - **Carpeta *Exercise2***:
    - ***Graph_ ex2.css***: estilos de las gráficas de todos los archivos del ejercicio 2.
    - ***GraphGrip_ ex2.js***: grafica los resultados obtenidos, para ambas manos, de la variable: **apertura máxima de pinza**. En el código JavaScript dichas variables son: *datos.opMaxPL* y *datos.opMaxPR* (apertura máxima de mano izquierda y derecha respectivamente).
    - ***GraphInd_ ex2.js***: grafica los resultados obtenidos, para ambas manos, de la variable: **rango máximo del dedo índice**. En el código

JavaScript dichas variables son: *datos.lirMax* y *datos.rirMax* (rango máximo del dedo índice izquierdo y derecho respectivamente).

- ***GraphPerimRing_ ex2.js***: grafica los resultados obtenidos, para ambas manos, de la variable: **perímetro del dedo anular**. En el código JavaScript dichas variables son: *datos.lringPer* y *datos.rringPer* (perímetro del dedo anular izquierdo y derecho respectivamente).
- ***GraphThumb_ ex2.js***: grafica los resultados obtenidos, para ambas manos, de la variable: **rango máximo del dedo pulgar**. En el código JavaScript dichas variables son: *datos.ltrMax* y *datos.rtrMax* (rango máximo del dedo pulgar izquierdo y derecho respectivamente).

- **Carpeta *Exercise3***:
  - ***Graph_ ex3.css***: estilos de las gráficas de todos los archivos del ejercicio 3.
  - ***GraphInd_ ex3.js***: grafica los resultados obtenidos, para ambas manos, de la variable: **rango máximo del dedo índice**. En el código JavaScript dichas variables son: *datos.lirMax* y *datos.rirMax* (rango máximo del dedo índice izquierdo y derecho respectivamente).
  - ***GraphPinky_ ex3.js***: grafica los resultados obtenidos, para ambas manos, de la variable: **rango máximo del dedo meñique**. En el código JavaScript dichas variables son: *datos.lprMax* y *datos.rprMax* (rango máximo del dedo meñique izquierdo y derecho respectivamente).
  - ***GraphRing_ ex3.js***: grafica los resultados obtenidos, para ambas manos, de la variable: **rango máximo del dedo anular**. En el código JavaScript dichas variables son: *datos.lringrMax* y *datos.rringrMax* (rango máximo del dedo anular izquierdo y derecho respectivamente).
  - ***GraphThumb_ ex3.js***: grafica los resultados obtenidos, para ambas manos, de la variable: **rango máximo del dedo pulgar**. En el código JavaScript dichas variables son: *datos.ltrMax* y *datos.rtrMax* (rango máximo del dedo pulgar izquierdo y derecho respectivamente).
- **Carpeta *Exercise4***:
  - ***Graph_ ex4.css***: estilos de las gráficas de todos los archivos del ejercicio 4.
  - ***GraphPerMax_ ex4.js***: grafica los resultados obtenidos, para ambas manos, de la variable: **perímetro máximo de la mano**. En el código JavaScript dichas variables son: *datos.leftPerMax* y *datos.rightPerMax* (perímetro máximo de la mano izquierda y derecha respectivamente).
  - ***GraphThumb_ ex4.js***: grafica los resultados obtenidos, para ambas manos, de la variable: **rango máximo del dedo pulgar**. En el código JavaScript dichas variables son: *datos.ltrMax* y *datos.rtrMax* (rango máximo del dedo pulgar izquierdo y derecho respectivamente).

- **Archivos de la carpeta *pages***: se encuentran los ficheros JavaScript que crean las vistas de cada ejercicio. La vista de cada ejercicio cuenta con una serie de botones que varían en cantidad y función dependiendo del ejercicio seleccionado.

Cuentan con la lógica programática, basada en el estado de ciertas variables, que permite enseñar la respuesta gráfica de un solo botón a la vez, evitando que la vista se llene de mucha información y sea poco comprensible. Además contiene el archivo CSS de los estilos que se aplican en los componentes de las vistas de cada ejercicio. Dichos archivos son:

- ***Ex_ page.css***: estilos de los componentes de las vistas de los ejercicios.
- ***Ex1_ page.js***: devuelve los botones y gráficas que se muestran en la vista del ejercicio 1.
- ***Ex1_ page.js***: devuelve los botones y gráficas que se muestran en la vista del ejercicio 2. .
- ***Ex1_ page.js***: devuelve los botones y gráficas que se muestran en la vista del ejercicio 3.
- ***Ex1_ page.js***:devuelve los botones y gráficas que se muestran en la vista del ejercicio 4.

Backend:

El *backend* posee prácticamente la misma estructura tanto en desarrollo local como dentro del servidor web de ROBOLABO. Todos sus archivos importantes están contenidos en una carpeta denominada *flask-server*. La única diferencia que tienen ambas formas de desarrollo es que en local se cuenta con una carpeta llamada *templates*, que se puede observar en el esquema de la Figura C.1, y que contiene un fichero *mostrar_ array.html* que se creó con la intención de visualizar las respuestas del servidor durante el desarrollo de la aplicación, ya que las aplicaciones de React, normalmente, no están diseñadas para visualizar respuestas que sean solamente variables númericas, *arrays* o listas de valores. por lo que se crean rutas de prueba dentro del fichero *server.py* que dirijan las respuestas a dicho HTML.

A continuación se describen las funciones de cada uno de los archivos contenidos en la carpeta *flask.server*:

- ***ConnectionDB.py***: es el fichero encargado de realizar la conexión con la base de datos y permitir recuperar información de la misma. Utiliza una función que recibe dos parámetros numéricos: *db_ num* y *coll_ num*. El primero de ellos puede tomar como valores 1 o 2, dependiendo de si se desea conectar con la *base de datos 1* o la *base de datos 2* respectivamente. El segundo parámetro toma valores desde 0 a 3 con el fin de configurar cual es la colección de la base de datos con la que se desea trabajar. Para la conexión con la base de datos que se encuentra en el servidor de ROBOLABO utiliza la siguiente URI: **mongodb://robolabocpu2:robolabo123@robolabo.etsit.upm.es:27017/admin**.
- ***login.py***: solo contiene un diccionario de valores con un usuario y una contraseña para el formulario del portal de acceso de la aplicación. El usuario es **userNDT** y la contraseña es **NDT2023**.
- ***patients.py***: se conecta con la base de datos mediante la función del fichero *ConnetionDB.py*, concretamente a la *base de datos 1* y a la colección *Usuarios*.

Realiza una busqueda de todos las personas que estan registradas en la base de datos para encontrar aquellas que en su campo *id* contengan la letra "p" indicando que son pacientes. Se crean dos listas, *usernames* y *ids*, para guardar la información deseada del paciente, que en este caso son el usuario y el *id* de cada uno. Devuelve como respuesta un diccionario de valores en el que están contenidas la lista de usuarios y la lista de *ids*.

- **Archivos *ex#.py***: son uno de los archivos que permiten el procesado de los datos y contienen una función que, recibiendo como parámetro el *username* del paciente, se conecta a la *base de datos 2* y a la colección *Estudios*. Encuentra el paciente especificado por el usuario y toma la información guardada en el campo *ejercicio#* de la base de datos y la guarda en una lista. Como los datos de los ejercicios están guardados en un *string*, hay que realizar un procesado de dichos datos para poder utilizarlos, y ese procesado se logra con los siguientes pasos:

  1. Dividir el *string* de los datos por "espacios en blanco" utilizando la función *split()* de Python.
  2. Separar la primera fila de valores (posición 0 de la lista) por "punto y coma" y guardar el resultado en una lista. De esta forma quedan todos los nombres de las variables que se miden.
  3. Con un bucle *for* sobre la lista del paso anterior se crean las claves del diccionario de valores (que son los nombres de las variables), que será la variable devuelta por esta función.
  4. Mediante dos bucles *for* anidados, se separan por "punto y coma" las filas restantes del *string* que contiene los datos del ejercicio y se adjunta cada resultado de la separación a una clave del diccionario. Esto logra que cada clave del diccionario se relacione con los valores de dicha variable.
  5. Utilizando otro bucle *for* se recorren los valores de cada clave del diccionario para reemplazar comas por puntos como los separadores decimales. Además, se convierten los valores de tipo *string* a tipo *float*.
  6. Se devuelve el diccionario de valores con las variables medidas por el dispotivo médico con sus respectivos valores.

  El caracter "#" del nombre se refiere a los valores que toma dicho caracter, desde el 1 al 4, para cada ejercicio.

- **Archivos *ex#_op.py***: son el otro archivo de cada ejercicio que permiten continuar y finalizar el procesado de datos comenzado en los archivos anteriores. Dicho procesado depende de cada ejercicio y consta de una serie de operaciones matemáticas para calcular las variables de interés que finalmente se grafican en el *frontend* de la aplicación. Estos archivos devuelven diccionarios de valores con las variables de interés y su valores, y estos diccionarios son los que recupera el *frontend* accediendo a las rutas virtuales del *backend*. El caracter "#" del nombre se refiere a los valores que toma dicho caracter, desde el 1 al 4, para cada ejercicio.

- ***server.py***: contiene las rutas virtuales del backend que se han desarrollado con Flask. Dichas rutas comprenden las que se utilizan para transferir información

del *backend* al *frontend* y las rutas de prueba comentadas anteriormente (están comentadas). Hay seis rutas utilizadas por la aplicación: una para la información de los pacientes (importa y utiliza la función del archivo *patients.py*), una para la información del inicio de sesión (importa y utiliza la función del *login.py*) y otras cuatro rutas para los ejercicios, donde cada una de esas rutas utiliza la función correspondiente de cada uno de los archivos del tipo *ex#_ op.py*. Las rutas para la información del paciente y del inicio de sesión solo aceptan peticiones HTTP de tipo GET, mientras que las rutas de los ejercicios aceptan peticiones HTTP de tipo GET y POST, utilizando esta última para poder recibir como parámetro el *username* del paciente que se desea visualizar en la aplicación. Dicho parámetro es utilizado en las funciones de los ficheros *ex#_ op.py* para obtener los resultados de un paciente específico y devolverlos como respues al *frontend* y graficarlos.

Consideraciones a la hora de agregar y/o editar estilos:

Para editar estilos basta con acceder a los archivos CSS que utilizan los componentes creados con React y editar los estilos que ya están establecidos o, incluso, se pueden agregar estilos si es lo deseado. A continuación se muestra una lista de todos los archivos CSS y su ubicación:

- ***App.css***: estilos aplicados en *App.js*. Ubicado en *src*.
- ***index.css***: estilos aplicados en *index.html*. Ubicado en *src*.
- ***Home.css***: estilos aplicados en *Home.js*. Ubicado en *src/pages*.
- ***Ex_ page.css***: estilos aplicados en los ficheros *Ex#_ page.js*. Ubicado en *src/pages*.
- ***compBS.css***: estilos aplicados en varios de los componentes de la interfaz de usuario. Son estilos de elementos de *react-bootstrap*. Ubicado en *src/components*.
- ***PatientsList.css***: estilos aplicados en los ficheros *PatientsList.js*. Ubicado en *src/components*.
- ***Graph_ ex1.css***: estilos aplicados en dos de los gráficos del ejercicio 1 indicados anteriormente, *GraphL_ ex1.js* y *GraphR_ ex1.js*. Ubicado en *src/components/Exercise1*.
- ***GraphBoth_ ex1.css***: estilos aplicados en uno de los gráficos del ejercicio 1 indicados anteriormente, *GraphBoth_ ex1.js*. Ubicado en *src/components/Exercise1*.
- ***Graph_ ex2.css***: estilos aplicados en todos los gráficos del ejercicio 2 indicados anteriormente. Ubicado en *src/components/Exercise2*.
- ***Graph_ ex3.css***: estilos aplicados en todos los gráficos del ejercicio 3 indicados anteriormente. Ubicado en *src/components/Exercise3*.
- ***Graph_ ex4.css***: estilos aplicados en todos los gráficos del ejercicio 4 indicados anteriormente. Ubicado en *src/components/Exercise4*.

Se debe tener especial atención si se edita el nombre de alguno de estos archivos ya que, en caso de ahcerlo, se deben cambiar también en las importaciones de los

componentes que utilizan dicho archivo. Si no, los estilos no podrán ser aplicados. Si lo que se desea es crear otro archivo CSS para algún componente ya existente o nuevo, basta con importarlo en dicho componente con el nombre escogido y colocar los estilos deseados dentro del CSS generado.

Consideraciones a la hora de agregar y/o editar componentes de la interfaz de usuario:

Los componentes se crean con los archivos de extensión *.js* y que importan React, desde la carpeta de dependencias, para poder utilizar la sintaxis JSX característica de React, que permite utilizar HTML y JavaScript en un mismo fichero. En caso de generar un nuevo componente, es clave realizar la importación de React y de cualquier dependencia o librería que se desee utilizar, asi como de los archivos de estilos. Otro paso clave para el uso de componentes es exportar dicho componente luego de cerrar la función que contiene dicho archivo, para ello se usa *export default nombre_ componente*.

Se recomienda que al agregar componentes nuevos se guarden en la carpeta *components*, con la excepción de aquellos componentes que engloban una serie de componentes para formar una vista completa, como es el caso de los ficheros *Ex#_ page.js* y que se almacenan en la carpeta *pages*. Además, se recomienda que el nombre de la función que contienen sea el mismo que el nombre del archivo o al menos relacionado y que empiecen por letra mayúscula, ya que es la forma de diferenciar elementos HTML de componentes React.

Al editar componentes ya existentes se debe tener especial cuidado con las direcciones de métodos *fetch()* o de importaciones de archivos, ya que el cambio erróneo de las mismas puede causar ejecuciones incorrectas de la aplicación o hasta imposibilitar su ejecución.

Como se menciona anteriormente, se deben identificar los componentes React con letras mayúsculas a la hora de crearlos para que cuando se importen desde otro componente y se incluyan en el código se puedan diferenciar las etiquetas que colocan componentes de las etiquetas que colocan elementos HTML.

En caso de querer agregar más rutas en el *frontend* se deben agregar en el archivo *App.js*, dentro de la variable *router*. Cada ruta debe ir acompañada del nombre de dicha ruta y el elemento que debe renderizar a la hora de acceder a ella. Dicho elemento son componentes React que, como se comenta previamente, engloban una serie de componentes para crear las vistas de la aplicación.

Consideraciones a la hora de hacer cambios en el *backend*:

Si se desean agregar más rutas virtuales al *backend* se deben colocar dentro del fichero *server.py* con la misma estructura que poseen las demás rutas que se pueden observar dentro de dicho fichero.

En el caso de la ejecución de la aplicación en el servidor de ROBOLABO, se utiliza la dependencia de Python denominada *waitress* para ejecutarla mediante un servidor de producción y no de desarrollo. Pero en caso de realizar desarrollo en local, se recomienda comentar las líneas de código respectivas y descomentar las líneas de código que se encuentran al final del fichero *server.py* para utilizar un servidor de desarrollo.

Si se desean visualizar resultados numéricos como *arrays*, listas, entre otros, se puede hacer accediendo a la dirección especificada del *backend* para dicho resultado. Algunas veces, por el tipo de variable o de operaciones que se realizan, este último método no funciona, por lo que se debe recurrir a las rutas virtuales de prueba que se encuentran comentadas al inicio del fichero *server.py*, que son rutas que renderizan una plantilla HTML donde se muestran resultados como *arrays* o listas de valores numéricos.

Por último, comentar el uso de cabeceras que permiten el uso de CORS en las peticiones HTTP que se realizan debido a que se comunican archivos de diferentes lenguajes y orígenes. CORS, por sus siglas en inglés *Cross origin resource sharing*, permiten el uso de información proveniente de un origen diferente al del navegador [51].

Consideración respecto al archivo *package.json*:

En el archivo *package.json* del *frontend*, se debe mencionar que en su contenido se encuentra un campo denominado “homepage” cuyo valor es la ubicación del servidor en donde se muestra la aplicación web desarrollada.

Apéndice D

Manual de usuario - Aplicación de visualización de datos

Este manual de usuario va dirigido a explicar los pasos básicos necesarios para interactuar con la interfaz de usuario de la aplicación web de visualización de datos que se presenta en este TFG. Como se explica en el Capítulo 2, los datos presentados en la aplicación provienen de un dispositivo médico llamado NeuroData Tracker utilizado en pruebas clínicas, en el Hospital Universitario La Paz, realizadas a pacientes con déficit motor.

Acceso a la aplicación:

Se puede acceder a la aplicación a través de la página web del dispositivo NeuroData Tracker (`http://www.robolabo.etsit.upm.es/NDTwork/NDT/NDT.html`) que posee una sección, al final de la página, con un botón que enlaza directamente con la aplicación de visualización de datos.

Figura D.1: Botón que redirecciona a la aplicación - Pagina web NDT

También se puede acceder directamente a través del siguiente enlace: `http://www.robolabo.etsit.upm.es/NDTwork/AppVisDatosNDT/frontend/build/`.

La primera vista que se tiene al ingresar en la aplicación es un portal de acceso que funciona con un único usuario y una única contraseña:

- **Usuario:** userNDT.
- **Contraseña:** NDT2023.

Después de introducir estas credenciales, se debe clicar en el botón "Iniciar sesión" para finalizar el proceso de acceso a la aplicación.

Figura D.2: Portal de acceso a la aplicación

Interacción con la interfaz de usuario:

Una vez dentro de la aplicación, la primera vista que se observa es una lista en la que se encuentran los pacientes identificados mediante su *id* correspondiente en la base de datos.

Cada elemento de esta lista es un desplegable que al clicar sobre el paciente muestra cuatro botones, cada uno correspondiente a un ejercicio de los cuatro que se realizan en las pruebas clínicas con el dispositivo. Se tomará como ejemplo al paciente que se identifica con el *id* "p103".

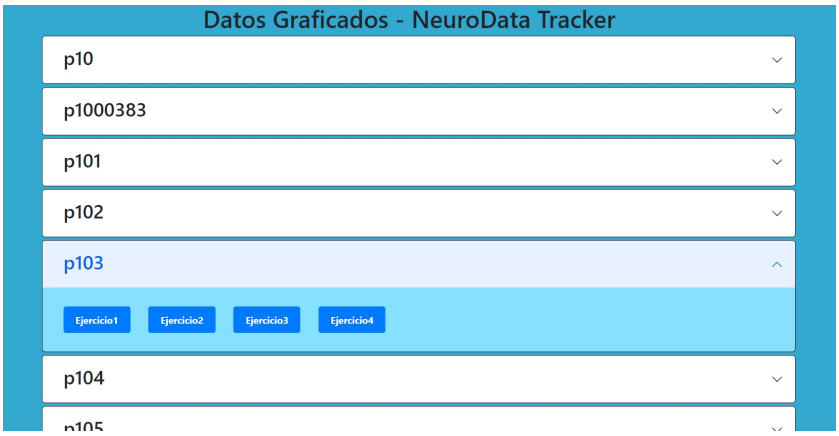


Figura D.3: Lista de pacientes con paciente **p103** seleccionado.

Al tener seleccionado el paciente deseado, se debe clicar sobre el botón del ejercicio que se quiere observar. Se recuerda que cada ejercicio cuenta con una vista propia que contiene una serie de botones, que varían en cada ejercicio, y que son los que permiten escoger el resultado que se desea observar. A continuación se presentan los ejercicios disponibles con sus respectivos botones:

- **Ejercicio 1:** flexo-extensión de muñeca.

Figura D.4: Botones del ejercicio 1.

- **Ejercicio 2:** pinza índice - pulgar.

Figura D.5: Botones del ejercicio 2.

- **Ejercicio 3:** separación de dedos.

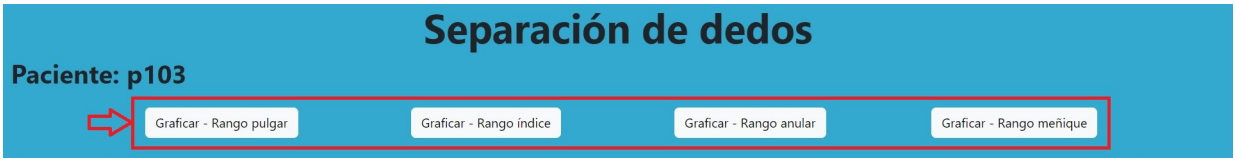


Figura D.6: Botones del ejercicio 3.

- **Ejercicio 4:** apertura - cierre de puño.

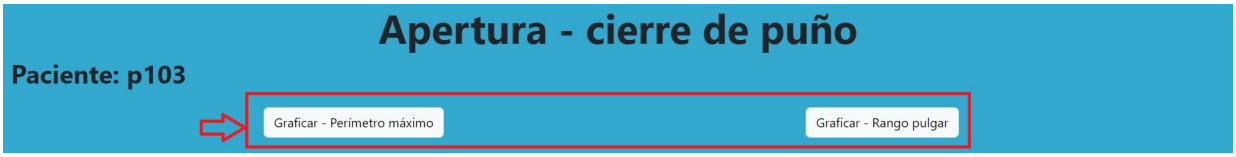


Figura D.7: Botones del ejercicio 4.

Consideraciones a tener en cuenta al interactuar con la aplicación:

- **Buscar pacientes en la lista:** ya que esta versión de la aplicación no cuenta con una barra de búsqueda implementada en su interfaz, se recomienda que para buscar un paciente, del cual se sabe su identificación (id), se utilice el comando de búsqueda del propio navegador. Dicho comando se activa al presionar las teclas **CTRL** + **F** al mismo tiempo.
- **Volver a la vista anterior:** al seleccionar la vista de cualquier ejercicio, la aplicación no posee botones para navegar a la vista anterior, la lista de pacientes. Para ello se recomiendda el uso de los botones del navegador, ubicados en la esquina superior izquierda de la pantalla, para realizar dicha tarea. Cada vez que se regrese a la lista de pacientes es posible que se deba buscar de nuevo al paciente ya que al volver a dicha vista se observa la lista desde el principio.
- **Algunos pacientes de la base de datos pueden dar errores:** en la lista se pueden encontrar pacientes de los cuales no se puedan visualizar los datos debido a los valores resultantes de los ejercicios realizados. Esto puede deberse a errores como mala ejecución del ejercicio, se dejó de captar las manos en algún momento del ejercicio o imposibilidad de completar el procesado de datos debido a valores no usuales que no permiten realizar ciertas operaciones matemáticas.

Apéndice E

Manual de desarrollador - Genereación automática de aplicaciones web estáticas

Este manual de desarrollador va dirigido a profundizar la estructura y funcionamiento del código que compone las plantillas del módulo de generación automática de aplicaciones web estáticas, con el fin de crear aplicaciones nuevas, editar las que ya existen o incluso mejorarlas.

Con la finalidad de contextualizar al lector, se debe aclarar que las aplicaciones web estáticas que se generan a partir de estas plantillas y sus recursos asociados poseen como idioma principal el inglés. Por ello, la información que se muestra en los archivos en todas las secciones se encuentra redactada en este mismo idioma, así como los nombres de ficheros y carpetas.

Primero se deben establecer los ficheros y carpetas para el funcionamiento de la aplicación:

- Un fichero HTML - *Project.html*.
- Un fichero CSS - *stylesProject.css*.
- Un fichero JS - *scriptsProject.js*.
- Un fichero JSON - *infoProject.json*.
- Una carpeta *Images* que contendrá las imágenes utilizadas en el sitio web.
- Una carpeta *videos_gif* que contendrá videos y/o gifs utilizados en el sitio web.

(Donde *Project* es el nombre o acrónimo del proyecto correspondiente. Se recomienda renombrar los ficheros de esta forma para mantener el orden e identificarlos sin dificultades)

Los ficheros nombrados anteriormente se pueden descargar desde la página web del Laboratorio de Robótica y Control, de la ETSIT-UPM, en el siguiente enlace: `http://www.robolabo.etsit.upm.es/NDT/templates/`. Para descargarlos correctamente se debe hacer clic con el botón derecho del ratón y pulsar la opción "*Guardar enlace*

como...". Se debe escoger el directorio donde se van a guardar dichos ficheros y, para finalizar, se pulsa el botón *Guardar*.

Para poder profundizar en el código que compone a las aplicaciones web estáticas que se están creando con este módulo, a continuación se comentará el contenido de cada fichero.

Fichero HTML:

En este fichero se encuentra la estructura de la plantilla de HTML. Las aplicaciones web creadas a partir de la misma pueden poseer todas o algunas de las secciones que se describen a continuación.

La estructura de la página web está configurada de tal manera que se garantice un comportamiento responsive (del término en inglés *responsive*). Esto debido a que se utiliza una gran variedad de elementos provenientes de Bootstrap y su hoja de estilos.

- **Metadatos**

  En la etiqueta <head> de los ficheros HTML se coloca información sobre los datos que contiene dicho archivo y la configuración de la página web, como título, codificación de caracteres y/o archivos externos requeridos.[52]

  En este caso se ha partido del archivo de una plantilla que ha sido editado y renombrado. Se ha utilizado la codificación de caracteres utf-8 que abarca una gran cantidad de caracteres y se han importado diferentes ficheros como los iconos de Bootstrap, el paquete de fuentes de Google, los estilos de SimpleLightbox y un último fichero de estilos de Bootstrap que es al que se han añadido los estilos creados y personalizados durante el desarrollo de la página web.

- **Cuerpo de la página - etiqueta <body>**

  La etiqueta <body> delimita el contenido de la página web que será visible para el usuario visitante [52]. Además, al final de esta etiqueta se importan los ficheros JavaScript que definen el comportamiento de los elementos de la aplicación. Concretamente se usan archivos JavaScript de Bootstrap, SimpleLightbox y el fichero por defecto de la plantilla. Este último contiene funciones preescritas por Bootstrap que se centran en el funcionamiento de la barra de navegación. Además, se ha renombrado y editado para agregar las funciones que crean la web automaticamente al comunicarse con el fichero JSON.

  - **Barra de navegación.**

    La primera sección que se encuentra dentro de la etiqueta <body> corresponde a la estructura de la barra de navegación, delimitada por la etiqueta <nav>. Esta consta, fundamentalmente, de dos elementos , uno para el logo y título, y el otro para las secciones de la página web. Estas

últimas se colocan como elementos de una lista no ordenada, delimitada por la etiqueta <ul>.

En el caso del logo y del título se utiliza la etiqueta <a> con la finalidad de que ambos elementos tengan la capacidad de redirigir al visitante al inicio de la página si se hace click sobre cualquiera de ellos.

- **Home/Inicio.**

  Es la primera sección de la aplicación web. Tiene una estructura sencilla basada en el sistem de rejilla de Bootstrap. Posee una fila y dos columnas. Una de las columnas va a mostrar una foto o gif, como fondo, que se establece en el fichero de estilos, específicamente en la clase nombrada *gifInicio*. La otra columna tiene el objetivo de mostrar una pequeña descripción introductoria del proyecto o herramiento que expone la página web.

  Esta sección tiene el objetivo de causar una buena primera impresión en los visitantes, por lo que se recomienda el uso de contenido multimedia llamativo y de textos cortos y concisos.

- **Motivation.**

  Es una sección destinada a dar a conocer el problema que se quiere resolver y lo que se plantea para abordarlo.

  En cuanto a su estructura, una parte de ella se basa en el sistema de rejilla de Bootstrap y la otra está compuesta por diferentes elementos como títulos <h1>, parrafos <p> y una barra divisora <hr>. A lo largo de la distribución de esta sección se colocan imágenes e incluso iconos, tomando en cuenta el paquete importado de Bootstrap del que se habló anteriormente.

  Se tiene un ejemplo en la figura C.1, con el fin de entender mejor la estructura. En ella se puede observar la sección de motivación de la página del proyecto *NeuroData Tracker* y los elementos multimedia que la conforman.

Figura E.1: Sección *Motivation* del proyecto NeuroData Tracker

- **Discovering.**

  En la sección *Discovering* se puede encontrar una descripción detallada sobre diferentes aspectos del proyecto. Esta formada por tres sub-secciones: *Characteristics*, *Project's parts* y *Functioning*.

  - **Characteristics.**

    Esta sección se basa en un elemento, denominado *Carousel*, predefinido por Bootstrap. Consiste en una galería de imágenes en donde se expone un texto, en este caso realcionado a las características de lo que se está exponiendo en la página.

    En dicha estructura se haya una gran variedad de elementos entre los cuales destacan los botones para navegar en la galería <button> y el elemento en donde se colocan los items correspondientes a las imágenes con sus respectivos textos, el cual se puede identificar por la clase llamada *carousel-inner*.

    En caso de que se quiera cambiar el color blanco del texto del *carousel* se puede cambiar al diseño oscuro de la siguiente manera:

    1. Añadir el texto *carousel-dark* a la clase del elemento que se encuentra en la linea 10 del código mostrado a continuación. Es decir, la clase de esa etiqueta debe ser: *class=çarousel slide carousel-dark"*.

    2. Editar todos los campos denominados *data-bs-target* y sustituir el valor *"#carouselExampleCaptions* por *"#carouselExampleDark"*.[53]

  - **Project's parts.**

Esta parte de la web tiene una estructura muy simple que se basa en el sistema de rejilla de Bootstrap. Además, cuenta con un elemento para el título de la propia sección como la mayoría de las otras secciones.

- **Functioning.**

  Esta sección posee una estructura destinada a albergar un video sobre el funcionamiento de la herramienta o proyecto en cuestión. Posee, en su mayoría, elementos configurados para que el video tenga un comportamiento responsivo y se adapte al tamaño de la pantalla.

- **Team.**

  Sección basada en el sistema de rejilla de Bootstrap. Contiene las imagenes y la información personal de los integrantes del equipo desarrollador del proyecto al que pertenece la paágina web.

- **Collaborators.**

  Sección basada en el sistema de rejilla de Bootstrap. Contiene los logos e información profesional sobre colaboradores que han prestado algún servicio o han aportado información.

- **Extra info.**

  En esta sección no hay una estructura definida ya que depende del proyecto al que este destinada la página web. En este caso tenemos el ejemplo de la página web del proyecto *NeuroData Tracker*, que consta de un video importado desde el canal de YouTube de uno de los colaboradores utilizando la etiqueta <iframe> y de una pequeña descripción sobre la información que se quiere transmitir con la etiqueta <p>.

  En otras aplicaciones web creadas a partir de estas plantillas se podría variar considerablemente la estructura de la sección dependiendo de los elementos que se deseen incluir para mostrar la información.

  En la siguiente imagen se puede ver el ejemplo expuesto en el código.

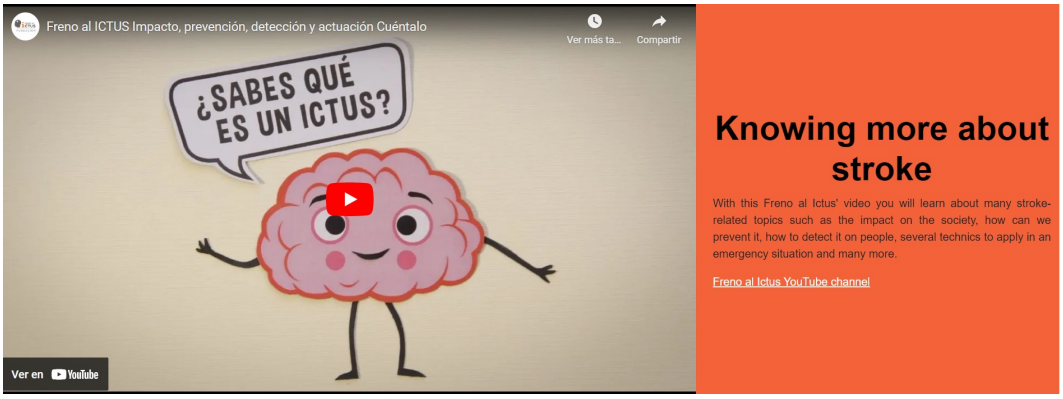


Figura E.2: Sección *Extra info* del proyecto NeuroData Tracker

- **Awards, media and publications/Premios, contenido y publicaciones.**

  Esta sección, al igual que la mayoría de las comentadas anteriormente, también se basa en el sistema de rejilla de Bootstrap y tiene el objetivo de albergar un contenido variado sobre reconocimientos, premios, noticias y/o algún otro tipo de información relevante sobre el proyecto, herramienta o servicio al que se orienta la página. Sus elementos pueden tener enlaces a páginas web externas donde se obtenga más información al respecto.

- **Footer/Pie de página.**

  Es la última sección de código visible y su estructura se puede dividir en 2 partes. Primero se observa un elemento destinado a las instituciones involucradas en los proyectos, donde se colocan los logos enlazados a las respectivas páginas web (se utilizan etiquetas tipo <img> y <a>.

  Por otro lado, se coloca una barra divisora <hr> para crear un espacio destinado a los derechos de diferentes imágenes utilizadas en secciones anteriores. En este caso, todas las imágenes que no son propias del proyecto provienen de la pagina web *Free Pik* y se colocan los enlaces a dicha web.

- **Enlaces a ficheros JavaScript**

  Es el último bloque de código que se coloca al final de la etiqueta <body> y contiene los enlaces a los scripts que definen el comportamiento de distintos elementos de la página web. En este caso se tienen scripts de Bootstrap, SimpleLightbox y el fichero JavaScript personalizado con las funciones que generan la aplicación.

  Se colocan en esta posición ya que los navegadores, a la hora de procesar los ficheros HTML, van por orden desde el inicio del documento hasta el

final. Y lo más apropiado es que luego de que se cargue la estructura y contenido, se procesen los ficheros JavaScript para evitar tiempos de carga de los mismos mientras no se ha mostrado ningún tipo de información.[54]

Fichero CSS:

El fichero CSS contiene los estilos de ids y clases utilizadas para configurar los elementos HTML de la aplicación web estática. Se divide en dos partes: los estilos predefinidos por Bootstrap y los estilos creados durante el desarrollo de la web que se añadieron al fichero.

- **Código CSS - Personalizado:**

  Se pueden encontrar estilos aplicados a diferentes ids y clases que se han utilizado a lo largo de todas las secciones. Al inicio del código se tienen ids correspondientes a los elementos de la barra de navegación para luego utilizarlos como enlaces y poder navegar haciendo click sobre estos elementos.

  Luego se tienen clases con la finalidad de configurar títulos, textos, dimensiones de imágenes o videos, fuentes y demás aspectos. Además, al final se tienen estilos especificados con *Media Query* para que la estructura HTML se comporte de manera responsiva, es decir, que se adapte al tamaño de la pantalla del dispositivo en donde se abre la aplicación web estática. Además, el sistema de rejilla de Bootstrap que se implementó en muchas de las secciones posee un conjunto de *breakpoints* que permiten profundizar en el diseño responsivo.

  Muchos de los estilos creados se identifican con nombres que se relacionan con la finalidad que tienen dentro del diseño de la página o a la sección en la que se aplican. Se pueden encontrar abreviaciones como *bg, pos, desc, img, mot* que significan *background, position, description, image and motivation* respectivamente.

  En el siguiente segmento de código se pueden observar los estilos aplicados según el dispositivo que esté usando el usuario al visitar la aplicación. Lo más importante a tener en cuenta es que la clase *gifInicio* solo se utiliza en dispositivos con pantallas de un tamaño mínimo de 992 píxeles, ya que para dispositivos más pequeños se decidió rediseñar el inicio de la página utilizando únicamente el título del proyecto y una descripción corta del mismo.

  Además, para dispositivos móviles (pantallas de máximo 768 píxeles) se redimensiona el tamaño del título de la barra de navegación así como el texto en la sección para la información del equipo (*sm-letters*. Por último se añade un elemento de relleno (*div-shrink*) en el inicio de la página para que la información se muestre correctamente. Para dispositivos con pantallas entre 769 y 991 píxeles solo se redimensiona el elemento de relleno que se añade al inicio.

```
/*Styles for different devices*/
/* For desktop: */
@media screen and (min-width: 992px){
  .col-1 {width: 8.33%;}
  .col-2 {width: 16.66%;}
  .col-3 {width: 25%;}
  .col-4 {width: 33.33%;}
  .col-5 {width: 41.66%;}
  .col-6 {width: 50%;}
  .col-7 {width: 58.33%;}
  .col-8 {width: 66.66%;}
  .col-9 {width: 75%;}
  .col-10 {width: 83.33%;}
  .col-11 {width: 91.66%;}
  .col-12 {width: 100%;}
  .gifInicio{
    background-image: linear-gradient(to left, rgb(255, 255,
        255,1), rgb(2,50,140,0)), url("videos_gif/Home.gif");
    background-repeat: no-repeat;
    background-position: center;
    background-size: cover;
    background-color: white;
    height: 713px;
    opacity: 1;
    position: relative;
  }
}

@media only screen and (max-width: 768px) {
  /*For mobile phones:*/
   #titleLogo{
    font-size: 90%;
   }

  .sm-letters{
    font-size: 75%;
  }

  .div-shrink{
    padding-top: 95px;
  }

  [class*="col-"] {
    width: 100%;
  }

}

@media screen and (min-width:769px) and (max-width:991px){
  /*For tablets*/
  .div-shrink{
    padding-top: 80px;
  };

}
```

Además de estas líneas de código, si se abre el fichero CSS, se observarán una serie de ids que no contienen ningún tipo de estilo aplicado. Esto se debe a que son ids que se utilizan para identificar elementos en la estructura de toda la página web y que se puedan editar con facilidad si así se desea.

- **Código CSS - Bootstrap:**

No se expone el código original de Bootstrap debido a que es una cantidad muy extensa de líneas. Por otro lado, a continuación se comentan algunas secciones de utilidad por si se desean cambiar colores, tamaños, fuentes, posición de los elementos o algún otro tipo de estilo.

- **Colores en la barra de navegación.**

Si se desea cambiar el color de las letras que representan las secciones, en la barra de navegación, se deben editar los colores hexadecimales de los estilos identificados como *.nav-item*. Para cambiar el color en el que se resaltan cuando el puntero está sobre las secciones se debe editar el estilo que se identifica en su parte final con *:hover*. En caso de querer cambiar el color del título principal en la barra de navegación se edita el otro bloque identificados como *.nav-brand* (tomando en cuenta la misma información mencionada anteriormente sobre el elemento *hover* pero para el título).

```
#mainNav.navbar-shrink .navbar-brand {
  color: #212529;
}
#mainNav.navbar-shrink .navbar-brand:hover {
  color: #c34e2e;
}
#mainNav.navbar-shrink .navbar-nav .nav-item .nav-link {
  color: #212529;
}
#mainNav.navbar-shrink .navbar-nav .nav-item .nav-link:hover
    {
  color: #f4623a;
}
```

- **Colores de enlaces.**

Si se desea cambiar el color en el que se resaltan los enlaces, en cualquier parte de la página, se debe cambiar el campo *color* dentro del estilo destinado a la etiqueta HTML *a*. Además, en este último también se puede cambiar la decoración de los enlaces. Por otro lado, si se desea editar el color en el que se iluminan los enlaces cuando el puntero se posiciona sobre los mismos, se debe editar el campo *color* del estilo *a:hover*.

```
a {
  color: #f4623a;
  text-decoration: underline;
```

```
}
a:hover {
  color: #c34e2e;
}
```

Nota: se recomienda, para obtener mejores resultados, que se consulten los valores hexadecimales o RGB de los colores que se desean utilizar en fuentes en línea.

Fichero JavaScript:

En este fichero se encuentran las funciones que generan parte de la estructura HTML de la aplicación web y que recuperan la información que se muestra en ella desde un fichero JSON. Esta última cualidad se logra debido al uso de una API con interfaz tipo JavaScript denominada *Fetch*, que permite obtener recursos de forma asíncrona a través de la red debido a un método global *fetch()* [55]. Este último es el utilizado en las funciones del fichero JavaScript para obtener la información del fichero JSON.

- **Funciones - Personalizadas:**

  - **Función *pageTitle()***

    Función que edita el HTML interno de la etiqueta <title> del bloque de metadatos del fichero HTML. Este título es el que se presenta en la pestaña en la que se accede a la aplicación web.

```
/* pageTitle is a function that edits the title inside <head>
    code */
function pageTitle(){
    fetch('infoProject.json')
        .then(res => res.json())
        .then(function title(res){
            const wpTitle = document.querySelector('#
                webPageTitle');
            wpTitle.innerHTML = res.project.title;
        })
}
```

  - **Función *createNavBar()***

    Función que crea la barra de navegación incluyendo el logo, título y secciones. Se basa en dos arreglos de datos denominados *li* y *a* que van a contener las secciones y los ids de cada una, respectivamente, para que se pueda explorar utilizando la barra de navegación. Posee un bucle *for* que recorre el array *wp_sections* y que contiene instrucciones para crear

los elementos HTML, con sus clases e ids, y que se colocarán en los *arrays* comentados anteriormente. Además, contiene un último bloque de código en donde se configura el logo y título, que se posicionan en el extremo izquierdo de la barra.

```
1  /* createNavBar is function that creates the navigation bar
       with all the web page's sections */
2  function createNavBar(){
3      fetch('infoProject.json')
4          .then(res => res.json())
5          .then(function sectionsNavBar(res){
6              //list and link HTML elements creation
7              let li = [];
8              let a = [];
9              const ul = document.querySelector('.navbar-nav')
10
11             //The following "for" is looping through the "
                   wp_sections" array
12             for(var i=0; i<res.project.sections.nav_bar.
                   wp_sections.length; i++){
13                 //Filling the list
14                 li[i] = document.createElement('li');
15                 list = li[i];
16                 list.setAttribute("class", "nav-item");
17
18                 //Setting the links
19                 a[i] = document.createElement('a');
20                 ref = a[i];
21                 ref.setAttribute("class", "navLinksDef");
22                 let set_href = "#" + res.project.sections.
                       nav_bar.wp_sections[i];
23                 ref.setAttribute("href", set_href);
24
25                 //Editing link inner html
26                 let html_a = res.project.sections.nav_bar.
                       wp_sections[i];
27                 let html_a_final = html_a.replace("_", " ");
28
29                 if(html_a_final == "Discovering"){
30                     html_a_final = html_a_final + " " + res.
                           project.acronym;
31                     ref.innerHTML = html_a_final;
32                 }else{
33                     ref.innerHTML = html_a_final;
34                 }
35                 //Setting the info inside the corresponding
                       elements
36                 list.appendChild(ref);
37                 ul.appendChild(list);
38             }
39
40             //Setting the logo
41             const logo = document.querySelector('#logo-project
                   ');
```

```
            let img = document.createElement('img');
            img.setAttribute("src", res.project.sections.nav_bar.logo)
            img.setAttribute("width", "43")
            img.setAttribute("height", "40")
            img.setAttribute("class", "d-inline-block align-text-top")
            logo.appendChild(img);
            const logo_title = document.querySelector('#titleLogo');
            logo_title.innerHTML = res.project.title;
        })
};
```

- **Función *createHome()***

  Función que crea la primera sección de la aplicación web. Es muy sencilla ya que solo se compone de instrucciones para crear elementos y configurarlos correctamente. Consta de dos elementos principales, *title* y *description*, que ambos son insertados en un elemento *div*.

```
/* createHome is a function that create the first section of the web */
function createHome(){
    fetch('infoProject.json')
        .then(res => res.json())
        .then(function home(res){
            //div, title and description HTML elements creation
            const div = document.querySelector("#title_description");
            let title = document.createElement('h1');
            let description = document.createElement('p');

            //Setting attributes and info
            title.setAttribute("class", "titles");
            description.setAttribute("class", "bbj-text");
            title.innerHTML = res.project.title;
            description.innerHTML = res.project.sections.home.description;
            div.appendChild(title);
            div.appendChild(description);
        })
};
```

- **Función *createMotivation()***

  En esta función solo se utilizan métodos como *querySelector()* e *innerHTML()* ya que la estructura de la sección está completamente colocada en el fichero HTML y con esta función solo buscamos recuperar la inforamción del fichero JSON e insertarla en las etiquetas de la estructura.

```
/* createMotivation is a function that creates the "Motivation
    " section. It doesnt use the method appendChild because
    this section has all the structure on the HTML file */
function createMotivation(){
    fetch('infoProject.json')
        .then(res => res.json())
        .then(function motivation(res){
            //Section's titles. The "for" is looping through "
                m_titles" array
            for(var i=0; i<res.project.sections.motivation.
                m_titles.length; i++){
                const m_title = document.querySelector('#
                    m_title_'+(i+1));
                m_title.innerHTML = res.project.sections.
                    motivation.m_titles[i];
            }

            //Setting the project logo
            const m_img = document.querySelector('#m_logo');
            m_img.setAttribute("src", res.project.sections.
                motivation.image);

            //Setting icons' titles. The "for" is looping
                through "m_titles_icons" array
            for(var i=0; i<res.project.sections.motivation.
                m_titles_icons.length; i++){
                const m_t_icons = document.querySelector('#
                    m_t_icons-'+(i+1));
                m_t_icons.innerHTML = res.project.sections.
                    motivation.m_titles_icons[i];
            }

            //Setting descriptions. The "for" is looping
                through "m_descriptions" array
            for(var i=0; i<res.project.sections.motivation.
                m_descriptions.length; i++){
                const m_desc = document.querySelector('#m_desc
                    -'+(i+1));
                m_desc.innerHTML = res.project.sections.
                    motivation.m_descriptions[i];
            }

        })
};
```

- **Función** ***createCharacteristics()***

  Es una función que tiene como objetivo principal la creación de los elementos que conforman la galería de imágenes que se utiliza en la subsección *Characteristics*. Además, cuenta con un bloque de código para la configuración del título de esta parte de la página web, al igual que muchas de las demás funciones.

Consta de un bucle *for* que recorre el arreglo de datos *charact* del fichero JSON. Dentro de este bloque se pueden encontrar cinco variables tipo *let* que están destinadas a contener la información. También consta de un condicionante de tipo *if* que se utiliza para configurar correctamente los elementos de la galería, ya que se basa en el id de dichos elementos y en caso de que este sea igual a uno (1), la clase atribuida al primer elemento de la galería es *carousel-item active*. En el caso del resto de elementos la clase atribuida es *carouse-item*. En caso de no cumplirse esta configuración no se podrán mostrar correctamente los elementos de la galería.

```
1  /* createCharacteristics is a function that creates the first
       part of "discovering" section (which is the second of the
       web page) */
2  function createCharacteristics(){
3      fetch('infoProject.json')
4          .then(res => res.json())
5          .then(function charact(res){
6
7              //Section's title
8              const div_ben = document.querySelector('#ben_t');
9              let title = document.createElement('h1');
10             title.setAttribute("class", "titles");
11             title.innerHTML = res.project.sections.discovering
                   .characteristics.title;
12             div_ben.appendChild(title);
13
14             //Selection of the element that will contain the
                   carousel items
15             const carousel = document.querySelector('.carousel
                   -inner');
16
17             //The following "for" is looping through the "
                   charact" array
18             for(var i=0; i<res.project.sections.discovering.
                   characteristics.charact.length; i++){
19
20                 //Creation of the HTML elements we need to
                       create the images' carousel
21                 let car_item = document.createElement('div');
22                 let img_item = document.createElement('img');
23                 let car_caption = document.createElement('div'
                       );
24                 let ben_title = document.createElement('h2');
25                 let ben_description = document.createElement('
                       p');
26
27                 //Setting the attributes of the previous
                       elements created
28
29                 //The following "if" is to make that the firts
                        carousel's item has the class "carousel-
                       item active"
30                 if(res.project.sections.discovering.
```

```
                    characteristics.charact[i].id == 1){
                     car_item.setAttribute("class","carousel-
                        item active");
                }else{
                     car_item.setAttribute("class","carousel-
                        item");
                }

                car_item.setAttribute("data-bs-interval","
                    10000");

                img_item.setAttribute("class", "img-fluid d-
                    block w-100");
                img_item.setAttribute("id", "carousel_img-" +
                    res.project.sections.discovering.
                    characteristics.charact[i].id);
                img_item.setAttribute("alt","Benefit " + res.
                    project.sections.discovering.
                    characteristics.charact[i].id);
                img_item.setAttribute("src", res.project.
                    sections.discovering.characteristics.
                    charact[i].image);
                //img_item.setAttribute("style", "height:565px
                    ; width:100%")

                car_caption.setAttribute("class", "carousel-
                    caption d-md-block");
                car_caption.setAttribute("id", "info_car-" +
                    res.project.sections.discovering.
                    characteristics.charact[i].id)

                //Setting the info inside the elements
                ben_description.setAttribute("class", "d-md-
                    block d-none")
                ben_title.innerHTML = res.project.sections.
                    discovering.characteristics.charact[i].
                    title;
                ben_description.innerHTML = res.project.
                    sections.discovering.characteristics.
                    charact[i].description;

                //Posting the info in the web
                car_caption.appendChild(ben_title);
                car_caption.appendChild(ben_description);

                car_item.appendChild(img_item);
                car_item.appendChild(car_caption);

                carousel.appendChild(car_item);
            }
        })
};
```

- **Función** ***createParts()***

Función basada en el sistema de rejilla de Bootstrap, por lo que se utilizan métodos para crear elementos tipo *row* y tipo *col*. Posee un bucle *for* que recorre el arreglo de datos *part* del fichero JSON. Dentro del mismo se hayan instrucciones para la creación de los elementos que conforman la rejilla y los elementos que se insertarán en ella para mostrar la información, configuración de dichos elementos y su inserción en la rejilla.

En este caso se tiene una variable *row* y dos variables que representan las dos columnas en las que se divide la sección que son *col_ img* y *col_ text*, que representan el espacio para las imágenes y para las descripciones respectivamente.

```
1  /* createParts is a function that creates the second part of "
       discovering" section (which is the second of the web page)
       . It is based on the use of bootstrap grid system */
2  function createParts(){
3      fetch('infoProject.json')
4          .then(res => res.json())
5          .then(function parts(res){
6              //Section's title
7              const div_parts_title = document.querySelector('#
                   div-parts');
8              let p_title = document.createElement('h1');
9              p_title.setAttribute("class", "titles");
10             p_title.innerHTML = res.project.sections.
                   discovering.parts.title;
11             div_parts_title.appendChild(p_title);
12
13             //Selection of the element that will have the grid
                   with all the content
14             const div_parts = document.querySelector('#
                   project_parts');
15
16             //The following "for" is looping through the "part
                   " array
17             for(var i=0; i<res.project.sections.discovering.
                   parts.part.length; i++){
18                 //Creation of the HTML elements we want to
                       create the grid, info and images
19                 let row = document.createElement('div');
20                 let col_img = document.createElement('div');
21                 let img = document.createElement('img');
22                 let col_text = document.createElement('div');
23                 let title = document.createElement('h2');
24                 let description = document.createElement('p');
25
26                 //Setting the attributes of the previous
                       elements created
27                 row.setAttribute("class", "row mgn-top-40");
28
29                 col_img.setAttribute("class", "col-lg-5 col-sm
                       -12 text-center");
30
```

```
                img.setAttribute("src", res.project.sections.discovering.parts.part[i].image);
                img.setAttribute("class", "img-fluid img-parts");

                col_text.setAttribute("class", "col-lg-7 col-sm-12 centro");

                title.setAttribute("class", "text-center");

                description.setAttribute("class", "parts-desc");

                //Setting the info inside grid's elements
                title.innerHTML = res.project.sections.discovering.parts.part[i].title;
                description.innerHTML = res.project.sections.discovering.parts.part[i].description;

                //Posting the info
                col_img.appendChild(img);

                col_text.appendChild(title);
                col_text.appendChild(description);

                row.appendChild(col_img);
                row.appendChild(col_text);
                div_parts.appendChild(row)
            }
        })
};
```

- **Función** ***createFunctioning()***

  Función encargada de cargar y configurar un video dentro de una estructura HTML que, en su mayoría, ya esta creada en el fichero HTML. Además, posee una sección que coloca el título de esta sección como en el caso de otras funciones.

```
/* createFunctioning is a function that creates the third, and last, part of "discovering" section (which is the second of the web page) */
function createFunctioning(){
    fetch('infoProject.json')
        .then(res => res.json())
        .then(function functioning(res){
            //Setting the video on the web page
            video = document.querySelector('.video');
            video.setAttribute("src", res.project.sections.discovering.functioning.video);

            //Section's title
            title = document.querySelector('#video_func');
            title_func = document.createElement('h1');
```

```
            title_func.setAttribute("class", "titles")
            title_func.innerHTML = res.project.sections.
                discovering.functioning.title;
            title.appendChild(title_func);
        })
};
```

- **Función *createDiscovering()***

  Esta función se encarga de llamar a las 3 funciones comentadas anteriormente: *createCharacteristics()*, *createParts()* y *createFunctioning*. Esto con la finalidad de que se generen las secciones respectivas de cada función.

```
/* createDescription is a function that creates all the
    sections composing "discovering" (Characteristics, Parts
    and Functioning). It calls each of the functions that
    creates each part */
function createDiscovering(){
            createCharacteristics();
            createParts();
            createFunctioning()
};
```

- **Función *createTeam()***

  Esta función posee un bloque principal de código compuesto por un bucle *for* que recorre el arreglo de datos *team_ members*. Dentro del mismo se encuentra un condicionante tipo *if* con la finalidad de que se lleve un conteo de las filas creadas, con la ayuda de la variable *row_ counter*. Esto permite que en la primera iteración se creee una fila y que este proceso se repita cada tres iteraciones.

  Se utilizan variables tipo *let* para la información que se inserta en la rejilla. Dichas variables son: *name*, *role*, *fltn*, *p_ link* y *link*.

```
/* createTeam is a fucntion that creates "Our team" section.
    It is based on the use of bootstrap grid system */
function createTeam(){
    fetch('infoProject.json')
        .then(res => res.json())
        .then(function team(res){
            //Section's title
            const team_title = document.querySelector('#
                team_title');
            let title = document.createElement('h1');
            title.setAttribute("class", "titles")
            title.innerHTML = res.project.sections.about_us.
                title;
            team_title.appendChild(title);

```

```
13            //Selection of the element that will have the grid
                 with all the members and their info
14            const team_grid = document.querySelector('#
                team_grid');
15
16            //row_counter is a variable that help us to count
                the number of rows created and to maintain the
                 symmetry of the grid. It is used in the
                following code sentences
17            var row_counter = 0;
18
19            //The following "for" is looping through "
                team_members" array
20            for(var i=0; i<res.project.sections.about_us.
                team_members.length; i++){
21                //Condition to create the rows every 3
                    iterations
22                if(i == 0 || i%3 == 0){
23                    row_counter++;
24                    row = document.createElement('div');
25                    row.setAttribute("class", "row")
26                    row.setAttribute("id", "row-"+(row_counter
                        ));
27                    team_grid.appendChild(row);
28                }
29
30                //Columns'creation and setting the info inside
                     them
31                col = document.createElement('div');
32                col.setAttribute("class", "col-xl-4 col-lg-12
                    col-sm-12 col-md-12 d-flex justify-content
                    -center");
33                col.setAttribute("id", "col-"+(i+1));
34
35                //Settng the image of each member
36                img = document.createElement('img');
37                img.setAttribute("id", "team_memb_img-"+(i+1))
                    ;
38                img.setAttribute("class", "margin-member-img")
                    ;
39                //img.setAttribute("style", "height: 250px;
                    width: 167px; margin: 20px;"); //borrar
                    luego que se haga el css
40                img.setAttribute("src", res.project.sections.
                    about_us.team_members[i].image);
41
42                //Setting the personal info of each member
43                info = document.createElement('div');
44                info.setAttribute("id", "info_member-"+(i+1));
45                info.setAttribute("style", "width: 100%;
                    height: 250px; margin: 20px;");
46                info.setAttribute("class", "sm-letters")
47
48                //Creating the HTML elements needed and
                    editing their inner HTML code
```

```
                let name = document.createElement('p');
                name.innerHTML = "<i class='bi bi-person'></i> " + res.project.sections.about_us.team_members[i].name;
                let role = document.createElement('p');
                role.innerHTML = "<i class='bi bi-file'></i> " + res.project.sections.about_us.team_members[i].role;
                let fltn = document.createElement('p');
                fltn.innerHTML = "<i class='bi bi-buildings'></i> " + res.project.sections.about_us.team_members[i].filiation;
                let p_link = document.createElement('p');
                p_link.innerHTML = "<i class='bi bi-link-45deg'></i> "
                let link = document.createElement('a');
                link.setAttribute("href", res.project.sections.about_us.team_members[i].info_link);
                link.innerHTML = "+info";

                //Posting the info
                p_link.appendChild(link);
                info.appendChild(name);
                info.appendChild(role);
                info.appendChild(fltn);
                info.appendChild(p_link);
                col.appendChild(img);
                col.appendChild(info);
                row.appendChild(col);
            }
        })
};
```

- **Función** ***createCollab()***

Función que se encarga de crear un sistema de rejilla muy parecido a la subsección *Parts* comentada anteriormente. Consta de un bucle *for* que recorre el arreglo de datos *collabs* del fichero JSON. Dentro de este bloque se tienen instrucciones para la creación de las filas y de las columnas, para ello se usan las variables *row*, *col_img* y *col_info*.

```
/* createCollab is a function that creates the "Collaborators" section. It is based on the use of bootstrap grid system */
function createCollab(){
    fetch('infoProject.json')
        .then(res => res.json())
        .then(function collab(res){

            //Section's title
            const collab_title = document.querySelector('#collab_title');
            let title = document.createElement('h1');
```

```
            title.setAttribute("class", "titles");
            title.innerHTML = res.project.sections.collaborators.title;
            collab_title.appendChild(title);

            //Selection of the collaborators section
            const collaborators_section = document.querySelector('#Collaborators');

            //Creating the grid and putting the info inside it
            //The following "for" is looping through "collabs" array
            for(var i=0; i<res.project.sections.collaborators.collabs.length; i++){
                //Creating and setting the div element which contains all the info about each collaborator
                let container = document.createElement('div');
                container.setAttribute("class", "container");
                container.setAttribute("id", "collabs_container")

                //Creating and setting the rows and columns of the grid
                let row = document.createElement('div');
                row.setAttribute("class", "row margin-collabs-row");

                let col_img = document.createElement('div');
                col_img.setAttribute("class", "col-lg-5 d-flex justify-content-center");

                let col_info = document.createElement('div');
                col_info.setAttribute("class","col-lg-7 centro");

                //Images with the respective link
                let img_link = document.createElement('a');
                img_link.setAttribute("id", "web_link_collab-" + (i+1));
                img_link.setAttribute("href", res.project.sections.collaborators.collabs[i].web_link);

                let img = document.createElement('img');
                img.setAttribute("class", "img-fluid");
                img.setAttribute("id", "img_collab-" + (i+1));
                img.setAttribute("src", res.project.sections.collaborators.collabs[i].image);

                //Posting the linked images
                img_link.appendChild(img);
                col_img.appendChild(img_link);
                row.appendChild(col_img);

```

```
                //Collaborators' name and info
                let collab_name = document.createElement('h2');
                collab_name.setAttribute("id", "collab_name-" + (i+1));
                collab_name.innerHTML = res.project.sections.collaborators.collabs[i].name;
                col_info.appendChild(collab_name);

                let collab_info = document.createElement('p');
                collab_info.setAttribute("class", "collabs-desc");
                collab_info.setAttribute("id", "collab_info-" + (i+1));
                collab_info.innerHTML = res.project.sections.collaborators.collabs[i].information;
                col_info.appendChild(collab_info);

                //Posting the info remaining
                row.appendChild(col_info);

                container.appendChild(row);

                collaborators_section.appendChild(container);
            }
        })
};
```

- **Función *createInfo()***

  Esta función solo se encarga de insertar la información del fichero JSON en la estructura que se encuentra en el fichero HTML. No crea elementos HTML de ningún tipo ya que la sección a la que está orientada posee una estructura que depende directamente de la información que se desee colocar. Por defecto, la plantilla está configurada para que esta sección albergue un video de YouTube y una pequeña descripción con el título respectivo. Para ello se cuenta con la configuración de un elemento *iframe* para que se pueda visualizar el video.

  En caso de que la información que se desee colocar en esta sección necesite otro tipo de elementos se debe de cambiar la estrucutura en el fichero HTML y editar las instrucciones de la función en el fichero JavaScript.

```
/* createInfo is a function that creates "Extra info" section */
function createInfo(){
    //This function doesn't require a loop.
    fetch('infoProject.json')
        .then(res => res.json())
        .then(function info(res){
            //Adding the link to the iframe video
```

```
            const iframe = document.querySelector('#iframe_video');
            iframe.setAttribute("src", res.project.sections.extra_info.video_link);

            //Adding the info (title, description and YT channel)
            const title = document.querySelector('#info_title');
            title.setAttribute("class", "titles")
            title.innerHTML = res.project.sections.extra_info.title;

            const video_info = document.querySelector('#video_info');
            video_info.innerHTML = res.project.sections.extra_info.video_info;

            const channel = document.querySelector('.linkYT');
            channel.setAttribute("href", res.project.sections.extra_info.channel_link);
            channel.innerHTML = res.project.sections.extra_info.channel_link_mask;
        })
};
```

- **Función *createAMP()***

  Función que crea un sistema de rejilla parecido al de la sección *Collaborators* y al de la subsección *Parts*. Para ello se tienen las variables *row*, *col_t* y *col_img*, donde las dos últimas representan las columnas para el texto y/o títulos y para las imágenes respectivamente. Consta de un bucle *for*, que recorre el arreglo de datos *content*, en donde se ejecutan las instrucciones correspondientes para la rejilla y para la creación e inserción de otros elementos en la misma. Además, cuenta con un condicionante tipo *if* que se encarga de verificar que el campo *link*, de la parte del fichero JSON correspondiente a esta sección, no sea una cadena de caracteres vacia. En caso de cumplirse esta condición se le añade un enlace a la iamgen insertada que lleva a un sitio web externo para obtener más información. En caso de que no se cumpla, solo se inserta la imagen.

```
/* createAMP is a function that creates "Awards, media & publications" section. It's based on bootstrap grid system */
function createAMP(){
    fetch('infoProject.json')
        .then(res => res.json())
        .then(function AMP(res){

            //Section's title
            const AMP_title = document.querySelector('#AMP_title');
```

```
9            let title = document.createElement('h1');
10           title.setAttribute("class", "titles");
11           title.innerHTML = res.project.sections.AMP.title;
12           AMP_title.appendChild(title);
13
14           //Selection of the div element which will have all
                 the content
15           const content_grid = document.querySelector('#
                 content_grid');
16
17           //The following "for" is looping through "content"
                 array
18           for(var i=0; i<res.project.sections.AMP.content.
                 length; i++){
19               //Creating the grid and info's elements:
20
21               //row
22               let row = document.createElement('div');
23               row.setAttribute("class", "row");
24
25               //content's title column
26               let col_t = document.createElement('div');
27               col_t.setAttribute("class", "col-lg-7 d-flex
                     align-items-center justify-content-center"
                     );
28
29               //content's image column
30               let col_img = document.createElement('div');
31               col_img.setAttribute("class", "col-lg-5 d-flex
                     justify-content-center");
32
33               //content's title
34               let content_t = document.createElement('h2');
35               content_t.setAttribute("id", "content_title");
36               content_t.innerHTML = res.project.sections.AMP
                     .content[i].title;
37
38               //content's image
39               let content_img = document.createElement('img'
                     );
40               content_img.setAttribute("class", "img-fluid
                     pub-size");
41               content_img.setAttribute("src", res.project.
                     sections.AMP.content[i].image);
42
43               //The following "if" creates an image linked
                     to a website in case the "link" field, in
                     the JSON file, is not empty
44               if(res.project.sections.AMP.content[i].link
                     !== ""){
45                   let img_link = document.createElement('a')
                         ;
46                   img_link.setAttribute("href", res.project.
                         sections.AMP.content[i].link);
47                   img_link.appendChild(content_img);
```

```
                    col_img.append(img_link);
                }else{
                    col_img.appendChild(content_img);
                }
                //Posting the info on the web:

                col_t.appendChild(content_t);
                row.appendChild(col_t);

                row.appendChild(col_img);

                content_grid.appendChild(row);

            }
        })
};
```

- **Función *createFooter()***

Función que crea el pie de página de la aplicación web. Se divide en dos partes, una de ellas para colocar el logo de las instituciones participantes y otra para los derechos de imágenes que se utilizan en la web.

Para la primera parte se cuenta con un un bucle *for* que recorre el arreglo de datos *footer_inst*. Dentro de esta estructura se tienen instrucciones para la creación de la columna correspondiente a cada logo de las instituciones y estos a su vez poseen un enlace al sitio web de cada institución. Para ello se utilizan las varaibles *col*, *a* y *img*. Todo ello insertado en una sola fila, creada en la variable *row*.

Para la segunda parte se recorre el arreglo de datos *img_credits* mediante un bucle *for* que se encarga de crear un elemento HTML con la etiqueta <a> para insertar los enlaces respectivos del sitio web donde se obtuvieron las imágenes, separados por un caracter /".

```
/* createFooter is a function that creates the web page's
    footer with the institutions' linked logos */
function createFooter(){
    fetch('infoProject.json')
        .then(res => res.json())
        .then(function footer(res){
            //The first part of the function is for the
                institutions
            //Selection of the div element which will contain
                de logos
            const row = document.querySelector('#institutions'
                );

            //"for" loop to create the columns and put the
                info inside them. It is looping through the "
                footer_inst" array
```

```
11              for(var i=0; i<res.project.footer_inst.length; i
                    ++){
12                  //Columns
13                  let col = document.createElement('div');
14                  col.setAttribute("class", "col");
15
16                  //Web pages' links
17                  let a = document.createElement('a');
18                  a.setAttribute("href", res.project.footer_inst
                        [i].web_link);
19
20                  //Images
21                  let img = document.createElement('img');
22                  img.setAttribute("class","imgsPP");
23                  img.setAttribute("src", res.project.
                        footer_inst[i].image);
24                  img.setAttribute("alt", "Logo" + res.project.
                        footer_inst[i].name);
25
26                  //Posting the info
27                  a.appendChild(img);
28                  col.appendChild(a);
29                  row.appendChild(col);
30              }
31
32              //The second part of the function is for image's
                    credits
33              //Selection of the p element that will contain
                    the credits' links
34              const credits = document.querySelector('.credits')
                    ;
35
36              //The following "for" is looping through "
                    img_credits" array
37              for(var i=0; i<res.project.img_credits.length; i
                    ++){
38                  //Creating the links
39                  let link = document.createElement('a');
40                  let separator = document.createElement('span')
                        ;
41                  separator.innerHTML = " / ";
42                  link.setAttribute("id","img_credit_" + res.
                        project.img_credits[i].id);
43                  link.setAttribute("href",res.project.
                        img_credits[i].link);
44                  link.innerHTML = res.project.img_credits[i].
                        link_mask;
45
46                  credits.appendChild(link);
47                  credits.appendChild(separator);
48              }
49
50          })
51  };
```

- **Función *createWebPage()***

  Última función del fichero JavaScript que se encarga de hacer un llamado a las funciones descritas anteriormente (excepto la de las subsecciones, que ya están incluidas en la función *createDiscovering()*) con el fin de crear la aplicación web en su totalidad. En la última linea de este fichero se hace un llamado a esta función para que actúe y se genere la página web.

```
/* createWebPage is a function that generates, automatically,
    all the sections of the web page. It calls all the
    previous functions (not bootstrap functions) */
function createWebPage(){
    pageTitle()
    createNavBar();
    createHome();
    createMotivation();
    createDiscovering();
    createTeam();
    createCollab();
    createInfo();
    createFooter();
    createAMP();
};

/* calling createWebPage to generate the web page */
createWebPage();
```

- **Funciones - Bootstrap:**

  El siguiente código JavaScript muestra las funciones que venían predefinidas en el fichero para la plantilla con la que se inicio la estructura de la aplicación. Se centran en el funcionamiento de la barra de navegación y la adaptabilidad de su comportamiento basado en el tamaño de pantalla del dispositivo en el que se inicia la aplicación.

```
/*!
* Start Bootstrap - Creative v7.0.6 (https://startbootstrap.com/
    theme/creative)
* Copyright 2013-2022 Start Bootstrap
* Licensed under MIT (https://github.com/StartBootstrap/
    startbootstrap-creative/blob/master/LICENSE)
*/
//
// Scripts
//

//Bootstrap functions:
window.addEventListener('DOMContentLoaded', event => {

    // Navbar shrink function
    var navbarShrink = function () {
        const navbarCollapsible = document.body.querySelector('#
            mainNav');
```

```
        if (!navbarCollapsible) {
            return;
        }
        if (window.scrollY === 0) {
            navbarCollapsible.classList.remove('navbar-shrink')
        } else {
            navbarCollapsible.classList.add('navbar-shrink')
        }

    };

    // Shrink the navbar
    navbarShrink();

    // Shrink the navbar when page is scrolled
    document.addEventListener('scroll', navbarShrink);

    // Collapse responsive navbar when toggler is visible
    const navbarToggler = document.body.querySelector('.navbar-
        toggler');
    const responsiveNavItems = [].slice.call(
        document.querySelectorAll('#navbarResponsive .nav-link')
    );
    responsiveNavItems.map(function (responsiveNavItem) {
        responsiveNavItem.addEventListener('click', () => {
            if (window.getComputedStyle(navbarToggler).display !==
                 'none') {
                navbarToggler.click();
            }
        });
    });
});
```

Fichero JSON:

Este fichero tiene la finalidad de albergar la información que se va a mostrar en la aplicación web estática. Se divide según las diferentes secciones que posee la página y la información se compone por pares *çampo":valor* donde el campo siempre debe de estar escrito entre comillas dobles y el valor puede tomar diferentes formas, entre ellas: *string*, *array*, *number*, *object* o *boolean*.

A continuación se encuentran los fragmentos en los que se divide la plantilla JSON del módulo de generación web. Se hace énfasis en la correcta escritura de los elementos a la hora de agregar o editar la plantilla, para ello se debe saber que los *string* deben escribirse entre comillas dobles, los *number* se escriben sin comillas, los *boolean* pueden tener como valor *true* o*false*, los *arrays* se declaran con corchetes ([ ]) y los *object* se declaran con llaves ({ }).

- **Nombre y acrónimo del proyecto:**

  Primeros campos que se deben proporcionar ya que serán utilizados para el título de la aplicación web y barra de navegación.

- **Barra de navegación:**

  Esta sección del fichero JSON ya posee un *array* de elementos tipo *string* que representan las secciones. Es editable en caso de que las secciones por defecto no sean las deseadas o se quiera añadir o eliminar secciones.

  Nota: La sección *Contact* se incluye en la barra pero no está creada. Se deja planteada para que se añada la información respectiva en un futuro.

- **Home:**

  Esta sección solo necesita que se proporcione una descripción corta del proyecto o herramienta.

- **Motivation:**

  Se propone que la motivación tenga dos títulos principales contenidos en el *array* denominado *m_titles*. Se tienen otros dos campos que representan subtítulos y pequeñas frases, respectivamente, que acompañan a iconos que forman parte de la estructura del fichero HTML y a los títulos. Por último se tiene un apartado denominado *image*, para que se coloquen los nombres de los archivos de imágenes que se utilicen en esta sección (siempre que estén dentro de la carpeta *Images*).

  Nota: Aunque los campos que poseen comillas dobles que están vacías se deben a que se recomienda dicho número de elementos, siempre es posible editar los campos teniendo en cuenta que se puede alterar la visualización de la información a la hora de cargar el sitio web..

- **Discovering:**

  - **Characteristics:**

    Para la información de las características se cuenta con los siguientes campos: *id*, *title*, *description* y *image*. El primero de ellos es para la identificación de los elementos. Luego se tienen dos campos que se deben rellenar con cadenas de caracteres para formar un título y una descripción que acompañen a la imagen propuesta en el último campo de los objetos que forman el *array* denominado *charact*.

    Nota: para agregar objetos al *array* para añadir más características se rocomienda copiar el bloque que se expone desde la línea 6 a la 11 y copiarlo seguidamente del otro, separándolos por una coma (",")..

  - **Parts:**

    Para la información de las partes se cuenta con los siguientes campos: *id*, *title*, *description* y *image*. El primero de ellos es para la identificación de los

elementos. Luego se tienen dos campos que se deben rellenar con cadenas de caracteres para formar un título y una descripción que acompañen a la imagen propuesta en el último campo de los objetos que forman el *array* denominado *part*. Además, se recomienda que en el campo *"title"* se sustituya el caracter "_"por el acrónimo o nombre del proyecto o herramienta.

Nota: para agregar objetos al *array* para añadir más partes se rocomienda copiar el bloque que se expone desde la línea 5 a la 10 y copiarlo seguidamente del otro, separándolos por una coma (",").

- **Functioning:**

  Solo hay dos campos, para el título y para el vídeo que se carga desde la carpeta *videos_gif*. Se recomienda que en el campo *"title"* se sustituya el caracter "_"por el acrónimo o nombre del proyecto o herramienta.

- **Team:**

Además del título de la sección, esta sección cuenta con un *array*, denominado *team_members*, que contiene objetos que representan a cada integrante del equipo. Cada uno posee los siguientes campos relacionados con la información de cada miembro: *id*, *name*, *role*, *filiation*, *info_link* y *image*.

Nota: para agregar objetos al *array* para añadir más integrantes se rocomienda copiar el bloque que se expone desde la línea 4 a la 11 y copiarlo seguidamente del otro, separándolos por una coma (",").

- **Collaborators:**

Además del título de la sección, esta sección cuenta con un *array*, denominado *collabs*, que contiene objetos que representan a cada colaborador que ha participado en el proyecto. Cada uno posee los siguientes campos de información: *id*, *name*, *image*, *information* y *web_link*.

Nota: para agregar objetos al *array* para añadir más colaboradores se rocomienda copiar el bloque que se expone desde la línea 4 a la 10 y copiarlo seguidamente del otro, separándolos por una coma (",").

- **Extra info:**

La información que se tiene en esta sección varía en base al proyecto al que se dirige la aplicación web. En este caso, la plantilla posee campos para albergar un título, enlaces a un video y canal de YouTube, un string para enmascarar el enlace y una descripción corta de lo que se esté exponiendo en esta parte de la página web. Se pueden agregar, eliminar o editar campos, pero se debe tener en cuenta que la estructura web de la plantilla está destinada para estos elementos.

- **Awards, media & publications:**

  Además del título de la sección, esta sección cuenta con un *array*, denominado *content*, que contiene objetos que representan a cada publicación. Cada uno posee los siguientes campos de información: *id*, *type*, *title*, *image* y *link*. El campo *type* es únicamente identificativo para cada publicación, es decir, tiene el objetivo de distinguir si es una publicación sobre una noticia, premio u otro tipo. El campo *link* puede rellenarse, o no, dependiendo de la publicación que se haga. En caso de que este campo no sea nulo, el enlace se coloca en la imagen respectiva.

  Nota: para agregar objetos al *array* para añadir más publicaciones se rocomienda copiar el bloque que se expone desde la línea 4 a la 10 y copiarlo seguidamente del otro, separándolos por una coma (`","`).

- **Contact:**

  La sección de contacto no contiene información porque es una parte que se plantea completar en futuras versiones de la aplicación y sus plantillas.

- **Footer:**

  Para el pie de página se tienen dos partes en su respectiva sección del fichero JSON. En la primera parte se alberga la información sobre las instituciones que se colocan en la web. Cada institución cuenta con un logo que está enlazado al sitio web respectivo. Además, poseen campos identificativos como lo son el *id* y *name*.

  Para la segunda parte se tiene la información correspondiente a los derechos de las imágenes que se han utilizado en la web, en caso de que no sean propias del proyecto que se está desarrollando. Cuenta con los siguientes campos: *id*, *link* y *link_mask*.

  Nota: para agregar objetos al *array* para se rocomienda copiar el bloque que se expone desde la línea 29 a la 33 y copiarlo seguidamente del otro, separándolos por una coma (`","`).

Subida de archivos en el servidor:

Todos las páginas web generadas a partir de estas plantillas y que estén relacionadas con los proyectos del Laboratorio de Robótica y Control (ROBOLABO) de la ETSIT-UPM se cargarán en el servidor de dicho laboratorio para que estén visibles al público. Para ello se explicarán los requisitos del servidor y pasos necesarios para cargar todos los archivos en el mismo.

- **Acceso al servidor:**

SSH es un protocolo de acceso remoto que está protegido criptográficamente [56]. Dicho protocolo es el que se utiliza para conectarse al servidor de ROBOLABO y requiere el uso de un usuario y una clave. Para ello se debe contactar con el profesor a cargo del laboratorio, Álvaro Gutiérrez, para que se generen tanto el usuario y como la clave. Se le puede contactar a través de su correo electrónico: a.gutierrez@upm.es.

Una vez se tiene conocimiento del usuario y clave, se debe instalar una aplicación como WinSCP (recomendada por el autor), OpenSSH, PuTTY, entre otras. En este caso se utilizó la aplicación WinSCP (solo disponible para Windiws) en un sistema operativo Windows 11, por lo que todos los pasos siguientes se desarrollan en dicho entorno:

1. **Descargar e instalar WinSCP:**

   Para descargar la aplicación WinSCP se accede al siguiente enlace y se hace click en el botón de descarga: `https://winscp.net/eng/download.php`. Cuando la descarga haya finalizado, se ejecuta el archivo y se completa la instalación.

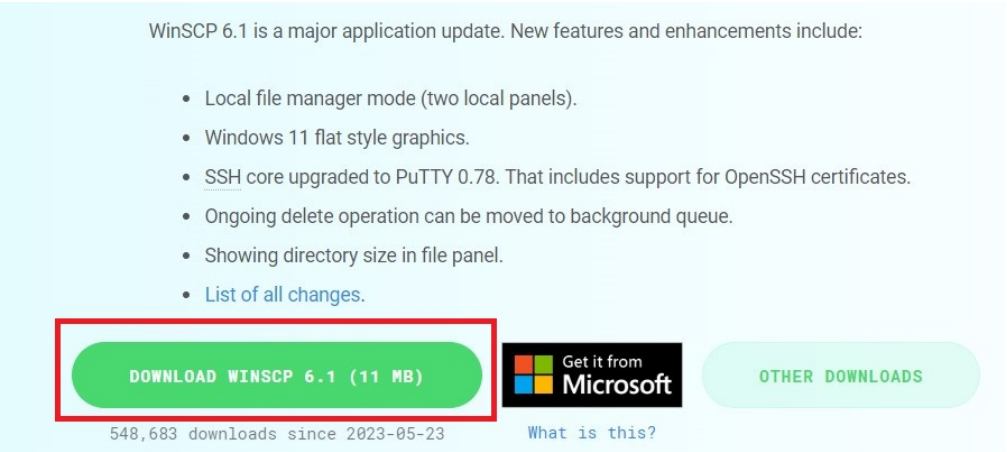


Figura E.3: Descarga de WinSCP

2. **Iniciar WinSCP y conexión con el servidor:** Al iniciar la aplicacion WinSCP seleccionamos el protocolo SCP y colocamos el nombre del servidor, que en este caso es: `http://www.robolabo.etsit.upm.es/`. Luego se introducen el usuario y la clave y se hace click en el botón *Conectar*.

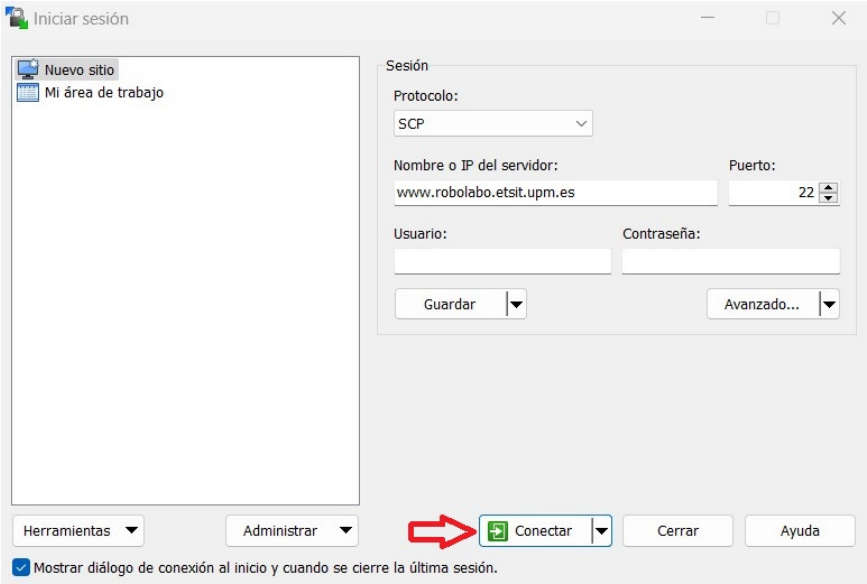


Figura E.4: Conexión con WinSCP

- **Subir archivos al servidor:**

Una vez se ha establecido la conexión con el servidor, se muestran dos paneles que muestran los directorios de nuestro dispositivo local (panel de la izquierda) y los de la dirección del servidor asignada para las páginas web (panel de la derecha).

Para cargar los archivos de un proyecto nuevo se debe buscar, en el panel de la izquierda, la ubicación de dichos ficheros en nuestro dispositivo y en el panel de la derecha, se debe asegurar que sea la ubicación, dentro del servidor, en donde se desea cargar los archivos. Luego se seleccionan los ficheros haciendo click sobre cada uno de ellos (para hacer una selección de múltiples archivos al mismo tiempo se debe usar la tecla *ctrl*) y se presiona el botón *Subir* de la barra superior de herramientas del panel izquierdo para cargarlos en el servidor.

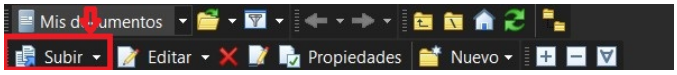


Figura E.5: Botón para cargar los archivos en el servidor

Si se desea actualizar los archivos de alguna página que ya haya sido cargada en el servidor anteriormente, se tienen dos opciones. La primera opción es eliminar los archivos que se quieran actualizar del servidor y repetir los pasos anteriormente explicados para la carga de ficheros. Para eliminarlos se utiliza un botón con una cruz roja en la barra de herramientas del panel derecho.

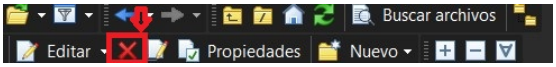


Figura E.6: Botón para eliminar los archivos del servidor

La segunda opción es cargar directamente los archivos en el servidor, y si los nombres del fichero actualizado en local y el fichero que se quiere reemplazar del servidor coinciden, se debe confirmar la opción de sobreescribir dicho fichero luego de presionar el botón de *Subir* mencionado anteriormente.

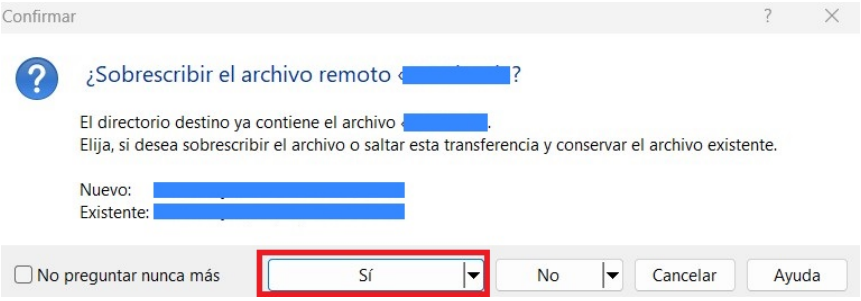


Figura E.7: Botón para sobreescribir los archivos del servidor

- **Requisitos del servidor en cuanto a los archivos:**

  Además de los requisitos de seguridad mencionados anteriormente, se deben tener en cuenta los siguientes aspectos en cuanto a la identificación de los ficheros:

  - Se recomienda que los nombres de los archivos no posean tildes.
  - Se debe ser muy cuidadoso al colocar los nombres de los ficheros ya que en el servidor se distingue entre mayúsculas y minúsculas.
  - Se recomienda que la extensión de los ficheros se coloque en minúsculas como un convenio para asegurar que todos los ficheros sean legibles.

Apéndice F

Manual de usuario - Generación automática de aplicaciones web estáticas

Este manual de usuario va dirigido a explicar los pasos básicos necesarios para que se genere una página web, aplicar diferentes estilos a elementos gráficos y/o editarlos. Para ello, el manual se dividirá en dos partes: generación de la página web y estilos de la página web.

Con la finalidad de contextualizar al lector, se debe aclarar que las páginas web que se generan a partir de estas plantillas y sus recursos asociados poseen como idioma principal el inglés. Por ello, la información que se muestra en los archivos en todas las secciones se encuentra redactada en este mismo idioma, así como los nombres de ficheros y carpetas.

Se debe comentar que, en caso de crear una página web o editarla, se debe poner en contacto con el profesor Álvaro Gutiérrez a través de su correo electrónico: a.gutierrez@upm.es, ya que es el encargado del Laboratorio de Robótica y Control de la ETSIT-UPM. Esto con el fin de enviar los archivos del proyecto para que se puedan cargar en el servidor una vez se haya finalizado la página web.

Primero se deben establecer los ficheros y carpetas para el funcionamiento de la aplicación:

- Un fichero HTML - *Project.html*.
- Un fichero CSS - *stylesProject.css*.
- Un fichero JS - *scriptsProject.js*.
- Un fichero JSON - *infoProject.json*.
- Una carpeta *Images* que contendrá las imágenes utilizadas en el sitio web.
- Una carpeta *videos_gif* que contendrá videos y/o gifs utilizados en el sitio web.

(Donde *Project* es el nombre o acrónimo del proyecto correspondiente. Se recomienda renombrar los ficheros de esta forma para mantener el orden e identificarlos sin dificultades)

Los ficheros nombrados anteriormente se pueden descargar desde la página web del Laboratorio de Robótica y Control, de la ETSIT-UPM, en el siguiente enlace: `http://www.robolabo.etsit.upm.es/NDT/templates/`. Para descargarlos correctamente se debe hacer clic con el botón derecho del ratón y pulsar la opción "*Guardar enlace como...*". Se debe escoger el directorio donde se van a guardar dichos ficheros y, para finalizar, se pulsa el botón *Guardar*.

Generación de la página web:

Para generar una página web, utilizando este módulo de generación automática, se debe colocar la información que se quiere mostrar en los campos del archivo *infoProject.json*, editando dicho archivo a través de un editor de texto o de código, como lo pueden ser el bloc de notas o Visual Studio Code respectivamente. En este caso, los pasos que se exponen a continuación se han realizado en el editor de código Visual Studio Code.

Además, se hace énfasis en la correcta escritura de la información sobre la plantilla para garantizar el funcionamiento adecuado de la misma. Para ello, se debe saber que la información que se coloque debe ir entre comillas dobles (). Por dicha razon, los diferentes campos de la plantilla ya están iniciados con comillas dobles en las que se debe colocar la información.

A continuación se exponen las secciones que se encuentran en el archivo *infoProject.json* y se explican los campos que componen cada una de ellas.

- **Título y acrónimo**

  El título y acrónimo del proyecto son los dos primeros campos que aparecen en el fichero. El título se utiliza para la sección *Home* y el acrónimo para la barra de navegación, concretamente a la hora de especificar la sección *Discovering*.

- **Barra de navegación y sección *Home***

  En cuanto a la barra de navegación, se tiene un campo que contiene las secciones de la página web por defecto. Si se desea eliminar alguna bastaría con borrar su nombre de dicho campo. Además, se debe comentar que la sección *Contact* se coloca en la barra de navegación. Por último, hay un campo denominado *logo* que posee el siguiente valor inicial:"Images". Teniendo en cuenta que el archivo del logo del proyecto debe guardarse en la carpeta *Images*, en este campo se debe colocar el nombre de dicho archivo luego de la barra y dentro de las comillas dobles.

  En cuanto a la sección *Home*, solo se tiene un campo denominado *description*

que debe contener una descripción breve del proyecto que irá acompañada del título mencionado anteriormente.

- **Sección *Motivation***

  En esta sección se tienen cuatro campos diferentes: *m_ titles*, *m_ titles_ icons*, *m_ descriptions* y *image*.

  - *m_ titles*: contiene dos títulos preestablecidos, uno para el planteamiento del problema que se aborda con el proyecto y el otro para presentar el proyecto como una solución a dicha problemática.

  - *m_ titles_ icons*: alberga el espacio para tres características que posicionan al proyecto como una buena solución.

  - *m_ descriptions*: se deben proporcionar cuatro breves descripciones. La primera de ellas debe ser sobre el problema que se aborda con el proyecto, mientras que las otras tres acompañarán a las características del campo anterior (deben ser aún más cortas que el plantemaiento del problema).

  - *image*: se debe colocar la ruta a la imagen del logo del proyecto, ya que se utiliza para presentar el proyecto como solución de una manera visual y llamativa. Dicha ruta debe ir entre las comillas y luego de la barra, ya que dicha imagen debe estar en la carpeta *Images*.

- **Sección *Discovering***

  La sección *Discovering* contiene tres subsecciones: *Characteristics*, *Parts* y *Functioning*. Cada una de ellas cuenta con un campo para el título y otros campos que se describen a continuación:

  - **Subsección *Characteristics***

    Su campo *charact* contiene las características que mejor describen el proyecto que se expone en la página web. Para ello, cada característica cuenta con una estructura definida por dos llaves ({ }) que se compone de cuatro campos: *id*, *title*, *description* y *image*.

    - *id*: se coloca un valor numérico que empieza en 1 apra la primera característica y va aumentando conforme agregamos características. Es un campo de identificación.
    - *title*: título de la característica.
    - *description*: breve descripción de la característica.
    - *image*: imagen que acompaña el texto formado por el título y la descripción mencionados anteriormente. Debe ser previamente guardada

en la carpeta *Images* y colocar su nombre en el campo, luego de la barra y entre las comillas.

Para agregar características diferentes se debe copiar el bloque de código comentado anteriormente que se encuentra entre llaves y pegarlo luego de la característica anterior, separando cada bloque por una coma. En el siguiente trozo de código se aprecia lo explicado:

```
                    "charact":[
                        {
                            "id":1,
                            "title":"",
                            "description":"",
                            "image":"Images/"
                        },
                        {
                            "id":2,
                            "title":"",
                            "description":"",
                            "image":"Images/"
                        }
                    ]
```

- **Subsección *Parts***

Su campo *charact* contiene las características que mejor describen el proyecto que se expone en la página web. Para ello, cada característica cuenta con una estructura definida por dos llaves ({ }) que se compone de cuatro campos: *id*, *title*, *description* y *image*.

  - *id*: se coloca un valor numérico que empieza en 1 para la primera parte y va aumentando conforme agregamos partes. Es un campo de identificación.
  - *title*: título de la parte.
  - *description*: breve descripción de la parte.
  - *image*: imagen que acompaña el texto formado por el título y la descripción mencionados anteriormente. Debe ser previamente guardada en la carpeta *Images* y colocar su nombre en el campo, luego de la barra y entre las comillas.

Para agregar partes se debe copiar el bloque de código comentado anteriormente que se encuentra entre llaves y pegarlo luego de la parte anterior, separando cada bloque por una coma. En el siguiente trozo de código se aprecia lo explicado:

```
                "parts":{
                    "title":"_ parts",

                    "part":[
```

```
5                           {
6                               "id":1,
7                               "title":"",
8                               "description":"",
9                               "image":"Images/"
10                          },
11                          {
12                              "id":2,
13                              "title":"",
14                              "description":"",
15                              "image":"Images/"
16                          }
17                      ]
18                  },
```

- **Subsección *Functioning***

  Su campo *video* tiene el siguiente valor inicial: "videos_gif". Aquí se debe colocar el nombre de un video que muestre el funcionamiento del dispositivo, aplicación o producto. Dicho nombre debe ir luego de la barra y entre las comillas dobles.

- **Sección *About us***

  Cuenta con un campo para el título de la sección y un campo *team_ members* que contiene la información de los integrantes del equipo, que se divide en otros seis campos: *id*, *name*, *role*, *filiation*, *info_ link* y *image*.

  - *id*: se coloca un valor numérico que empieza en 1 para la primera parte y va aumentando conforme agregamos partes. Es un campo de identificación.
  - *name*: nombre y apellidos del integrante.
  - *role*: rol que cumple la persona en el equipo.
  - *filiation*: institucion a la que pertenece.
  - *info_ link*: enlace a pagina web personal o LinkedIn.
  - *image*: imagen personal que debe ser previamente guardada en la carpte *Images*. El nombre del archivo debe colocarse luego de la barra, entre las comillas dobles.

  Para agregar miembros se debe copiar el bloque de código comentado anteriormente que se encuentra entre llaves y pegarlo luego del miembro anterior, separando cada bloque por una coma. En el siguiente trozo de código se aprecia lo explicado:

```
1                   "team_members":[
2                       {
3                           "id":1,
4                           "name":"",
```

```
                        "role":"",
                        "filiation":"",
                        "info_link":"",
                        "image":"Images/"
                    },
                    {
                        "id":2,
                        "name":"",
                        "role":"",
                        "filiation":"",
                        "info_link":"",
                        "image":"Images/"
                    }
                ]
```

- **Sección *Collaborators***

Cuenta con un campo para el título de la sección y un campo *collabs* que contiene la información de los colaboradores del proyecto, que se divide en otros cinco campos: *id*, *name*, *iamge*, *information* y *web_link*.

- *id*: se coloca un valor numérico que empieza en 1 para el primer colaborador y va aumentando conforme agregamos colaboradores. Es un campo de identificación.
- *name*: nombre de la persona, empresa u organización colaboradora.
- *image*: imagen o logo que represente al colaborador
- *information*: descripción del colaborador y sus acciones dentro del proyecto.
- *web_link*: enlace a página web, canal de YouTube o algún otro recurso web que reresente al colaborador y se pueda visitar para obtener más información.

Para agregar colaboradores se debe copiar el bloque de código comentado anteriormente que se encuentra entre llaves y pegarlo luego del colaborador anterior, separando cada bloque por una coma. En el siguiente trozo de código se aprecia lo explicado:

```
                "collabs":[
                    {
                        "id":1,
                        "name":"",
                        "image":"Images/",
                        "information":"",
                        "web_link":""
                    },
                    {
                        "id":2,
                        "name":"",
                        "image":"Images/",
                        "information":"",
```

```
                        "web_link":""
                    }
                ]
```

- **Sección *Extra info***

  Esta sección es para incorporar información complementaria que sea de interés para entender la problemática que se está abordando y contextualizar de una manera más completa al visitante del sitio web. Por defecto contiene el espacio para colocar un video acompañado de un breve párrafo relacionado. Está compuesta de cinco campos:

  - *title*: título de la sección. Por defecto este campo tiene como valor inicial: "Knowing more about...", con el fin de que se sustituyan los tres puntos suspensivos por el tema complementario que se trata en esta sección.
  - *video_link*: enlace al video de YouTube.
  - *channel_link*: enlace al canal de YouTube del creador del video.
  - *channel_link_mask*: se recomienda escribir en este campo una pequeña frase que sirva para guiar al visitante al canal de YouTube. Por ejemplo: Çanal de YouTube de Fundación Freno al ICTUS". Esto va a sustituir, visualmente, al enlace del canal.
  - *video_info*: breve descripción sobre lo que se trata en la sección y el video. Se debe colocar entre las comillas dobles y debe ir antes del elemento *
* que se encuentra en la plantilla. Este último es para garantizar una estructura adecuada.

- **Sección *Awards, media & publications (AMP)***

  En esta sección se coloca información referente a publicaciones relacionadas al proyecto. Para ello, cuenta con el título de la sección y un campo *content* donde se coloca la información de dichas publicaciones repoartida en cinco campos:

  - *id*: se coloca un valor numérico que empieza en 1 para la primers publicación y va aumentando conforme agregamos publicaciones. Es un campo de identificación.
  - *type*: se indica el tipo de publicación. Puede ser una noticia, publicación de redes sociales, reconocimientos, premios o alguna otra categoría. Es un campo de identificación.
  - *title*: título de la publicación.
  - *image*: imagen relacionada con la publicación.
  - *link*: enlace a la publicación, que será utilizado para que al hacer click en la imagen se redireccione a dicha publicación. Si no hay, se deja en blanco.

Para agregar publicaciones se debe copiar el bloque de código comentado anteriormente que se encuentra entre llaves y pegarlo luego de la publicación anterior, separando cada bloque por una coma. En el siguiente trozo de código se aprecia lo explicado:

```
                "content":[
                    {
                        "id":1,
                        "type":"",
                        "title":"",
                        "image":"Images/",
                        "link":""
                    },
                    {
                        "id":2,
                        "type":"",
                        "title":"",
                        "image":"Images/",
                        "link":""
                    }
                ]
```

- **Sección *Contact***

  Esta sección no está configurada todavía, pero se coloca para tenerla en cuenta en futuras actualizaciones del módulo de generación automática de aplicaciones web estáticas.

- **Pie de página (*footer*)**

  El pie de página se divide en dos partes: las instituciones principales y los créditos a imágenes utilizadas provenientes de algún sitio web de internet (solo si es el caso).

  Para la parte de las instituciones no se debe agregar información ya que la plantilla ya cuenta con ella. Las instituciones involucradas son la Escuela Técnica Superior de Ingenieros de Telecomunicación (ETSIT), el Laboratorio de Robótica y Control de la ETSIT, IdiPaz y el Hospital Universitario La Paz.

  Para la parte de los derecho de imagen, en caso de ser necesaria, se cuenta con un campo *img_ credits* en el que encontramos otros tres campos para completar la información necesaria de la imagen recopialda de internet.

  - *id*: se coloca un valor numérico que empieza en 1 para la primers publicación y va aumentando conforme agregamos publicaciones. Es un campo de identificación.
  - *link*: enlace que redirecciona al visitante a la página web origen de esa imagen.

- *link_ mask*: texto que enmascara el enlace de la imagen. Por ejemplo, si tomamos la imagen del sitio web "Freepik.es", se puede poner lo siguiente: Ïmagen1 - Imagen de Freepik.º si tiene algun autor específico dentro del sitio web, se pondría Ïmagen1 - Imagen de Usuario en Freepik". Donde *Imagen1* es un identificador de la imagen dutilizada dentro de la página web y *Usuario* es el nombre del usuario creador de dicha imagen.

Estilos de la página web:

En esta parte del manual se explicarán los pasos necesarios para poder modificar algunos elementos básicos de los estilos que se aplican a la página web y cambiar la estética de la misma. Para ello, se debe abrir el archivo *stylesProject.css* con un editor de texto o de código, como lo pueden ser el bloc de notas o Visual Studio Code respectivamente. En este caso, los pasos que se exponen a continuación se han realizado en el editor de código Visual Studio Code.

- **Colores en la barra de navegación.**

  Si se desea cambiar el color de las letras que representan las secciones, en la barra de navegación, se deben editar los colores en hexadecimal de los estilos identificados como *.nav-item*. Para cambiar el color en el que se resaltan cuando el puntero está sobre las secciones se debe editar el estilo que se identifica en su parte final con *:hover*. En caso de querer cambiar el color del título principal en la barra de navegación se edita el otro bloque identificados como *.nav-brand* (tomando en cuenta la misma información mencionada anteriormente sobre el elemento *hover* pero para el título).

```
#mainNav.navbar-shrink .navbar-brand {
  color: #212529;
}
#mainNav.navbar-shrink .navbar-brand:hover {
  color: #c34e2e;
}
#mainNav.navbar-shrink .navbar-nav .nav-item .nav-link {
  color: #212529;
}
#mainNav.navbar-shrink .navbar-nav .nav-item .nav-link:hover {
  color: #f4623a;
}
```

- **Colores de enlaces.**

  Si se desea cambiar el color en el que se resaltan los enlaces, en cualquier parte de la página, se debe cambiar el campo *color* dentro del estilo destinado a la etiqueta HTML *a*. Por otro lado, si se desea editar el color en el que se iluminan los enlaces cuando el puntero se posiciona sobre los mismos, se debe editar el campo *color* del estilo *a:hover*.

```
a {
  color: #f4623a;
  text-decoration: underline;
}
a:hover {
  color: #c34e2e;
}
```

- **Color de fondo.**

  El color de fondo por defecto es naranja y se coloca en el estilo *bg-orange*. Para cambiarlo se debe modificar el código hexadecimal del color y se recomienda cambiar el nombre del estilo sustituyendo .ºrange"por el nombre del color deseado en inglés. Luego se debe abrir el archivo HTML de la página web, utilizar el comando de búsqueda mencionado anteriormente, y buscar "bg-orange"para encontrar las tres localizaciones de ese estilo y cambiar la palabra .ºrange"por el color escogido. De esta manera el fichero HTML se comunica correctamente con el fichero CSS y viceversa.

- **Gif de la sección *Home*.**

  En la sección Home se recomienda colocar un GIF que se relacione con el proyecto. Para ello, este se debe guardar previamente en la carpeta *videos_gif*.

  UNa vez guardado el GIF, se debe utilizar el comando de búsqueda mencionado anteriormente y buscar "gifInicio", de esta forma se encuentra el estilo que controla el GIF mostrado en la primerea sección de la página. Una vez se encuentra, hay que modificar la url que aparece en el campo *background-image* y sustituir el nombre "Home.gif"por el nombre del GIF que se quiere colocar.

- **Iconos de la sección *Motivation*.**

  En la sección *Motivation* se utilizan iconos que acompañan a las características que se exponen. Dichos iconos pueden varias dependiendo de las características que se coloquen.

  Para cambiar esos iconos debemos visitar la página web de iconos de Bootstrap: `https://icons.getbootstrap.com/`. Una vez dentro, se buscan los iconos que se desean colocar y se seleccionan. Al seleccionar un icono, aparece una línea de código a la derecha, de la cual solo debemos copiar el valor escrito entre comillas dobles del campo *class*. Luego debemos abrir el archivo HTML de la página web y, utilizando el comando de búsqueda mencionado anteriormente, buscar “i ” (la letra “i” más un espacio en blanco), lo que llevará a la posición de esos iconos. Se puede observar que cada etiqueta referente a los iconos se ve de la siguiente forma: *<i class=“CLASE-ICONO fs-1 icons-Color”>*, donde “CLASE-ICONO” se refiere a la clase que debemos editar pegando el valor copiado de la pagina web de Bootstrap sustituyendo el que está actualmente en la plantilla. Se

hace énfasis en que solo se debe cambiar esa parte del campo *class* de la etiqueta HTML.

Se recomienda utilizar el comando de búsqueda de palabras para encontrar rapidamente los estilos y/o etiquetas mencionados anteriormente. En el caso del bloc de notas se presionan las teclas ctrl+B, mientras que en Visual Studio Code se presionan las teclas ctrl+F. Además, se debe tener en cuenta que a la hora de cambiar colores en esta plantilla se pueden utilizar tanto códigos hexadecimales como códigos RGB.

Se debe recordar que al realizar cualquier cambio deseado en alguno de los ficheros mencionados anteriormente, se debe guardar dicho archivo para que los cambios se puedan visualizar. Para guardar el archivo se puede hacer a través de la barra de herramientas de la aplicación que haya usado para abrir los ficheros, concretamente el apartado "Archivo" o "File", y luego pulsar "Guardar" o "Save". También se puede hacer más fácilmente con el comando rápido de guardado, pulsando las teclas ctrl+S.

Nota: se recomienda, para obtener mejores resultados, que se consulten los valores hexadecimales o RGB de los colores que se desean utilizar en fuentes en línea.