UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y
SERVICIOS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

DISEÑO E IMPLEMENTACIÓN DE UN SIMULADOR CLÍNICO WEB INTERACTIVO PARA EL TRATAMIENTO DE TRAUMATISMOS

LUIS CASTAÑEDA LÓPEZ

2020

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y
SERVICIOS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

DISEÑO E IMPLEMENTACIÓN DE UN SIMULADOR CLÍNICO WEB INTERACTIVO PARA EL TRATAMIENTO DE TRAUMATISMOS

Autor
LUIS CASTAÑEDA LÓPEZ

Tutora
BLANCA LARRAGA GARCÍA

2020

Resumen

En general, en cualquier ámbito profesional o incluso personal es recomendable tener experiencia previa ante cualquier situación. Si se habla del ámbito clínico, adquirir esa experiencia previa ante una situación de emergencia es difícil y en ocasiones, tan sólo se adquiere enfrentándose a situaciones de ese tipo a lo largo de la carrera profesional. Esto puede acarrear equivocaciones o retrasos en situaciones que no lo permiten, viéndose afectada la salud de una persona. Es por ello que los simuladores clínicos sirven precisamente para ganar experiencia y aprender de los errores, preparando a los profesionales médicos a afrontar cualquier situación de emergencia con amplias garantías de éxito en cada decisión tomada.

Algunas de las situaciones graves que se pueden encontrar los profesionales médicos son los traumatismos. Los traumatismos graves suponen una altísima morbilidad cada año en cuanto a la población mundial se refiere, destacando los traumatismos causados por autolesiones y por accidentes de tráfico. En muchos de estos incidentes, un traumatismo grave no supone la muerte, pero sí implica múltiples secuelas y discapacidades que acompañarán a la víctima a lo largo de su vida. Estos sucesos requieren de una atención inmediata y efectiva. El cuidado que se hace de estos pacientes en las primeras horas son esenciales para salvar su vida y evitar futuras secuelas que afecten a su calidad de vida.

Este Trabajo Fin de Grado surge de la unión de las dos preocupaciones mencionadas anteriormente. La idea es diseñar un simulador clínico que ayude a los profesionales médicos a adquirir experiencia, así como aprender y conocer los diferentes protocolos de actuación ante lesiones producidas por un traumatismo grave. Se trata de una herramienta didáctica que emula un caso clínico traumático específico en el que el profesional o el estudiante pueda aprender, mejorar y adquirir nuevas habilidades de manera práctica que le ayuden a estar preparado ante cualquier situación real similar.

Para alcanzar el objetivo propuesto se ha desarrollado una aplicación web interactiva para el tratamiento de lesiones traumáticas. El proceso inicia con el estudio de las diferentes lesiones traumáticas y de los protocolos de actuación ante cada una de ellas. En cuanto a la aplicación, consiste en un servidor donde se alojan los distintos casos clínicos planteados, así como los protocolos de actuación adaptados a las diferentes interacciones que se realizan en la parte visual de la que también consta la aplicación, donde se ve representada la simulación propuesta. Todo ello adaptado para conseguir un despliegue del simulador sencillo y rápido en cualquier lugar. Lo que permitirá el aprendizaje de los distintos protocolos y maniobras que debe realizar un profesional ante cualquier traumatismo grave de una manera rápida y eficiente.

Palabras clave: Traumatismo, simulación, protocolo, aprendizaje, Javascript, frontend, backend, React, Node.js, Express, Sequelize, MySQL, Docker.

Abstract

Overall, in any professional or even personal environment it is advisable to have previous experience in any situation. If we talk about the clinical area, acquiring this previous experience in an emergency situation is difficult and sometimes it is only acquired by dealing with similar situations during their professional career. This can lead to mistakes or delays in situations that do not allow it, affecting to people's health. That is why clinical simulators are used, to get experience and learn from mistakes, training medical professionals to deal with any emergency situation with a high guarantee of success in each decision adopted.

Some of the serious situations that medical professionals may have to face are traumas. Critical traumas represent a very high morbidity illness every year, highlighting traumas caused by self-injuries and traffic accidents. In many of these incidents, a critical trauma does not cause death, but it does mean multiple consequences and disabilities that will affect the patient for the rest of the life. These events require immediate and effective attention. The care given to these patients in the first few hours is essential to save their lives and to avoid future consequences affecting their life quality.

This End-of-Degree Project is the result of the union of the two concerns mentioned above. The idea is to design a clinical simulator that helps medical professionals to acquire experience, as well as to learn and know the different protocols of action in the case of injuries produced by critical trauma. It is a didactic tool that emulates a specific trauma case in which the professional or the student can learn, improve and acquire new skills in a practical way. This will help the clinicians to be prepared for any similar situation.

To achieve the proposed objective, an interactive web application has been developed for the treatment of traumatic injuries. The process begins with the study of the different traumatic injuries and the action protocols for each of them. Regarding the application, it consists of a server where the different clinical cases are stored, as well as the action protocols adapted to the different interactions that are carried out in the visual part of the application, where the proposed simulation is represented. All of this is adapted to achieve a simple and fast deployment of the simulator in any place. This will allow to learn the different protocols and techniques that a professional must perform to treat critical traumatisms in a fast and efficient way.

Keywords: Trauma, simulation, protocol, learning, Javascript, frontend, backend, React, Node.js, Express, Sequelize, MySQL, Docker.

Agradecimientos

Antes de comenzar, me gustaría dar las gracias a las personas que me han ayudado en el desarrollo de este Trabajo Fin de Grado.

En primer lugar, a mis tutores Blanca y Álvaro por darme la oportunidad de realizar este trabajo, y por ayudarme siempre que lo he necesitado.

A mis amigos, por el apoyo y la ayuda recibida desde el principio.

Y por último, a mis padres y a mi hermana, gracias a los cuales he podido estudiar lo que me gusta, contando siempre con su apoyo.

GRACIAS

Índice general

Índice de figuras

Índice de tablas

Lista de Acrónimos

HULP: Hospital Universitario La Paz.

OMS: Organización Mundial de la Salud.

UE: Unión Europea.

INE: Instituto Nacional de Estadística.

SEMICYUC: Sociedad Española de Medicina Intensiva, Crítica y Unidades Coronarias.

UCI: Unidad de Cuidados Intensivos.

ATLS: Advanced Trauma Life Support.

GCS: Glasgow Coma Scale.

TAC: Tomografía Axial Computarizada.

UML: Lenguaje Unificado de Modelado.

MVC: Modelo-Vista-Controlador.

DOM: Document Object Model.

NPM: Node Package Manager.

API: Aplication Programming Interface.

SQL: Structured Query Language.

ORM: Object Relational Mapping.

PK: Primary Key.

FK: Foreign Key.

COIT: Colegio Oficial de Ingenieros de Telecomunicación.

Capítulo 1

Introducción

Este Trabajo de Fin de Titulación ha sido realizado en colaboración y con la supervisión del Hospital Universitario La Paz (HULP), con el fin de desarrollar un simulador clínico interactivo que sea capaz de entrenar a los profesionales ante diferentes escenarios.

Los traumatismos graves precisan de una atención especializada que requiere de gran coordinación y rapidez en la atención al paciente para que los daños sufridos no sean irreparables o incluso mortales. Por lo tanto, el simulador interactivo aparece como una alternativa para poder formar al personal clínico a hacer frente a estas situaciones así como a las posibles consecuencias que cada actuación clínica pueda tener en el paciente.

1.1. Enfermedad traumática grave

Una enfermedad traumática grave es aquella cuyas lesiones son causadas por un traumatismo que puede poner en riesgo la vida de una persona, además de la posibilidad de que aparezcan discapacidades importantes.

La Organización Mundial de la Salud (OMS) considera al trauma grave como una de las 10 principales causas mortales y de discapacidad, siendo la principal causa de muerte para los menores de 45 años, lo que supone el 12 % de las muertes globales. El accidente de tráfico se destaca como el principal responsable de esta fatalidad. En la Figura 1.1 se puede ver que los accidentes de tráfico son la octava causa principal de muertes en el mundo.
Por su parte, la Unión Europea (UE) publicó en 2016 un estudio que ubica a los accidentes en el quinto lugar del ranking de las principales causas mortales en Europa tal y como se muestra en la Figura 1.2.

Aunque la gran mayoría de estas muertes provocadas por un trauma se producen en accidentes de tráfico, y especialmente entre la población más joven; destacan también las precipitaciones y las caídas accidentales, principalmente en personas mayores [9].

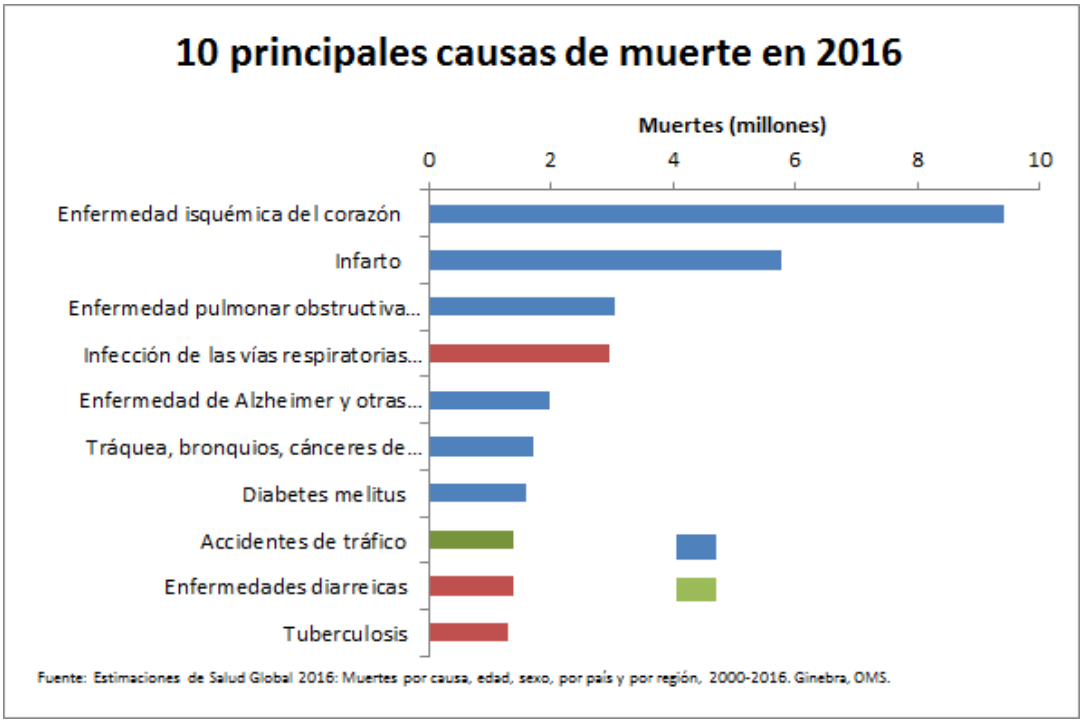

Figura 1.1: Principales causas de muerte en el mundo.[1]

Según el Instituto Nacional de Estadística (INE), en el año 2018 se produjeron en España 427.721 muertes en total, de las cuales 15.768 ocurrieron por causas externas, sumando el 3,7 % del total [10]. Las causas externas pueden clasificarse o desglosarse en accidentes como la autolesión o el suicidio, que ocupan el primer lugar. Seguido de la caída accidental y el ahogamiento. En cuarto lugar aparecen los accidentes de tráfico, que agrupan el mayor porcentaje de las muertes por causas externas en la población más joven, como refleja la Figura 1.3.

Según el estudio de la SEMICYUC, la edad media de los pacientes traumatizados que ingresan en la UCI es de 47 años, siendo en su mayoría varones. Alrededor de la mitad de los ingresados requieren una intervención quirúrgica en las primeras 24 horas y aproximadamente tres cuartas partes precisan de respiradores. Además, la tasa de mortalidad por traumatismo grave se sitúa en torno al 15 % [11].

Adicionalmente y por causa del envejecimiento de la población, el número de traumatismos graves aumenta en personas mayores de 65 años, donde la evolución de estos puede ser más complicada, ya que a menudo es posible que padezcan otras enfermedades o necesiten ciertos medicamentos que dificultan el tratamiento y su posterior recuperación del traumatismo. Es por esto que los traumatismos tienen el doble de letalidad en los ancianos que en los grupos más jóvenes [9].

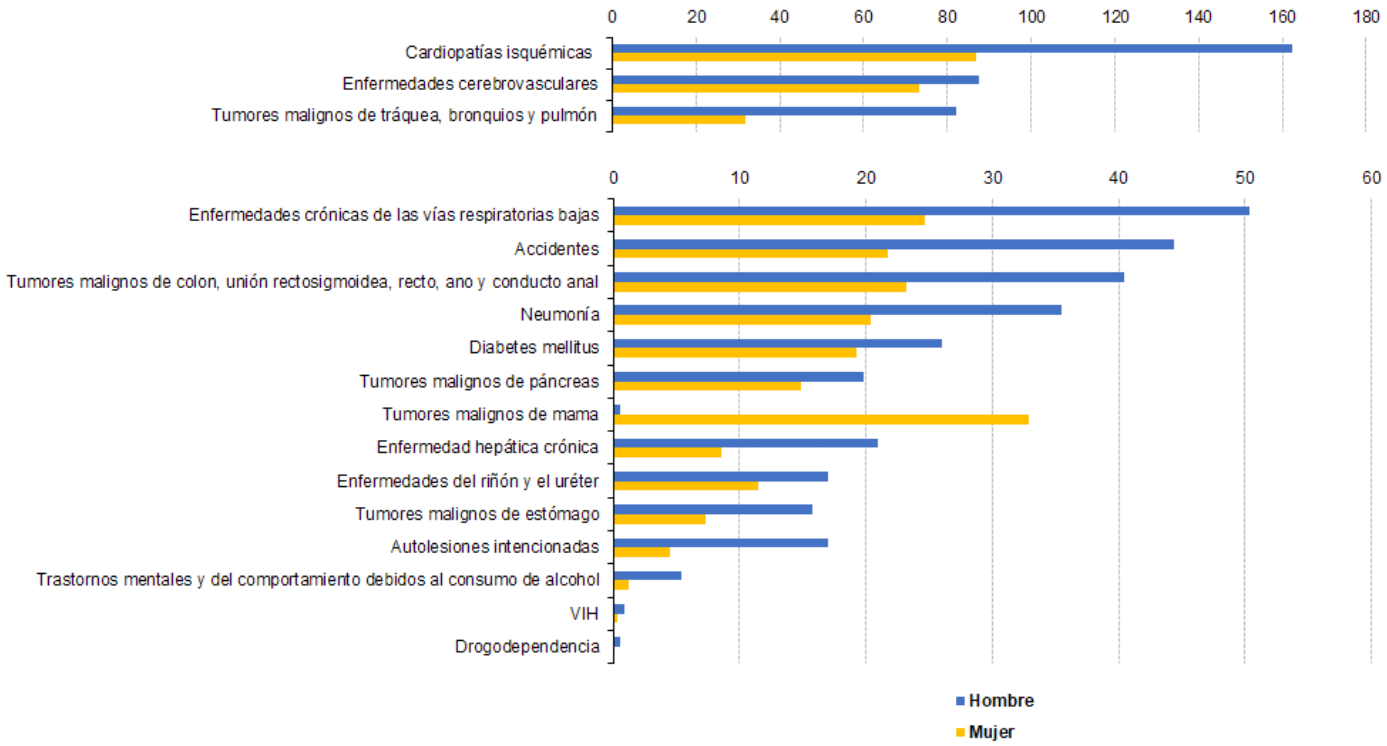

Figura 1.2: Causas mortales (por 100.000 hab.), EU, 2016. [2]

Los traumatismos más frecuentes son de tipo craneoncefálico, seguido del traumatismo torácico, del traumatismo en extremidades y del traumatismo abdominal.
El traumatismo craneoencefálico posee la mayor tasa de mortalidad y es el que acarrea las secuelas más importantes [9]. Estos traumatismos conllevan lesiones neurológicas y peligrosas hemorragias, presentes en la mayoría de las muertes tempranas, que se producen en los primeros minutos tras el incidente.

La letalidad de la enfermedad del trauma grave se puede clasificar en tres etapas [12]:

- La primera tiene lugar en el momento que se produce el accidente. Los motivos de una muerte rápida pueden ser lesiones graves en el sistema nervioso central, paros cardiacos o hemorragias graves.
- La segunda etapa discurre desde los primeros minutos hasta las horas posteriores al traumatismo. Estas muertes se deben a obstrucciones de la vía aérea, traumatismos craneocenfálicos, neumotórax, hemoperitoneo o fractura pélvica entre otras complicaciones.
- La tercera y última etapa se produce días o incluso semanas después de que se produciera el accidente, y normalmente ocurren por sepsis o por disfunción orgánica múltiple.

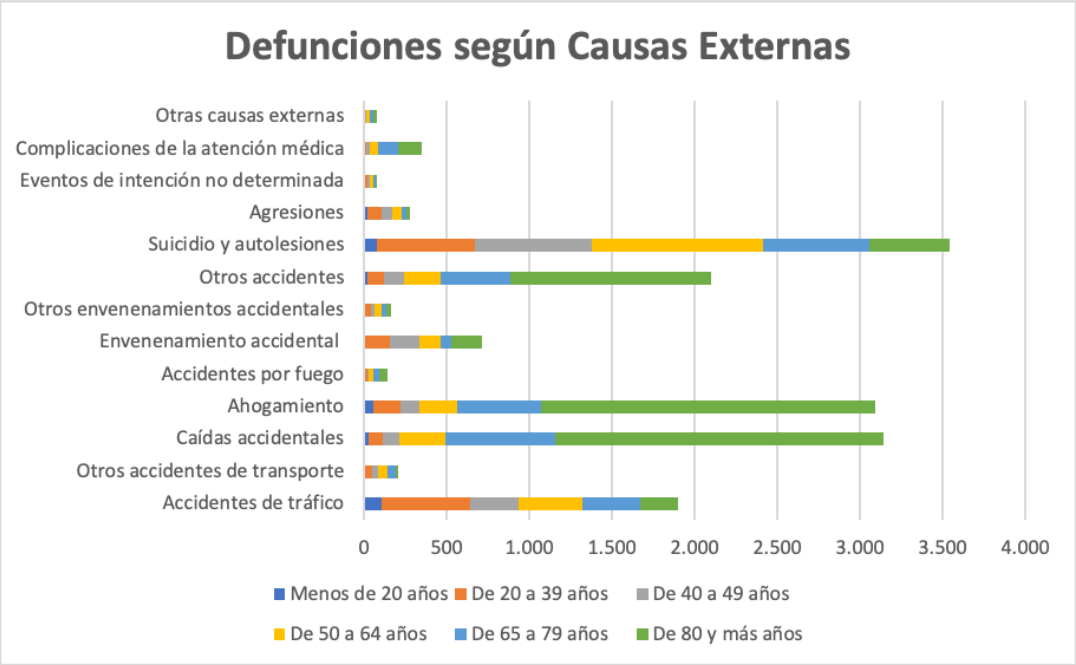

Figura 1.3: Defunciones debido a causas externas , INE, 2018 [3].

La evolución del paciente depende del tipo de traumatismo y de la energía del impacto. Adicionalmente, la reserva fisiológica del paciente también es importante puesto que puede haber pacientes que tengan enfermedades y tratamientos previos que pueden complicar la evolución del trauma. Este trabajo surge de la necesidad de mejora continua del tratamiento a pacientes que sufren un traumatismo gracias al entrenamiento de los profesionales por medio de un simulador clínico. Este simulador va a ayudar a desarrollar las habilidades de los profesionales en la aplicación de los cuidados a los pacientes con este tipo de lesiones. Los simuladores se utilizan en el ámbito clínico para formar a profesionales clínicos en el diagnóstico y en la evaluación de las lesiones y las maniobras pertinentes a realizar. Estas maniobras se deben realizar en el menor tiempo posible, con el fin de salvar vidas y disminuir tanto el número de víctimas mortales como la gravedad de posibles secuelas o discapacidades [12].

1.2. Simulador clínico

Actualmente el mundo está profundamente influenciado por la tecnología. Así pues, la medicina ha evolucionado en paralelo con los avances de la tecnología para desarrollar y mejorar las capacidades de sus profesionales. Una muestra de este avance son los simuladores clínicos. Estos simuladores emulan el comportamiento de una situación real para el entrenamiento de habilidades y competencias, preparando a los profesionales clínicos para el momento en el cual se enfrenten a esas situaciones reales.

La simulación clínica consiste en un conjunto de métodos que facilitan a los estudiantes la adquisición de habilidades y destrezas clínicas, en escenarios parecidos a los reales, sin poner en riesgo la vida de las personas, utilizando pacientes virtuales. El aprendizaje virtual tiene como principales características la inmaterialidad, la interactividad, la autonomía y la digitalización. En este proceso educativo, la interacción entre el estudiante y el docente cuenta con la ayuda de la tecnología [13].

Hay diferentes tipos de simuladores utilizados para mejorar las aptitudes de los profesionales y realizar una evaluación crítica analizando la situación representada en la simulación y evaluando las acciones llevadas a cabo, como pueden ser [14]:

- **Emuladores de baja tecnología:** simuladores que replican una parte del organismo permitiendo el desarrollo de habilidades psicomotoras básicas.
- **Pacientes simulados**: se trata de actores preparados para actuar como pacientes.
- **Emuladores virtuales en pantalla:** programas informáticos que permiten el entrenamiento y evaluación de los conocimientos y las decisiones adoptadas. Se suelen utilizar en campos como la fisiología y la farmacología.
- **Emuladores de tareas complejas:** son dispositivos electrónicos, computacionales y mecánicos con altas funcionalidades visuales y táctiles que permiten una representación tridimensional. Con frecuencia, estos simuladores son combinados con emuladores de baja tecnología para fusionar la interacción física con el entorno virtual, permitiendo el desarrollo de habilidades manuales.
- **Emuladores de pacientes completos:** se trata de maniquíes de tamaño real manejados digitalmente simulando aspectos anatómicos y fisiológicos. También permiten gestionar situaciones complejas y mejorar habilidades como el trabajo en grupo.

También, los simuladores pueden ser clasificados en función de su fidelidad. Esta fidelidad se refiere al grado de realismo que poseen los simuladores utilizados. Así pues, las simulaciones se dividen en baja, media y alta fidelidad [15].

- **Baja fidelidad:** son simuladores anatómicos en los que se practican maniobras invasivas y no invasivas.
- **Fidelidad intermedia:** surge de la combinación de una parte anatómica y de una parte software que permiten estudiar ciertos aspectos médicos del paciente.
- **Alta fidelidad:** simuladores hardware y software que representan la fisiología de un paciente virtual consiguiendo un alto realismo en las simulaciones, ilustrando situaciones clínicas de todo tipo.

Al igual que en otras ramas de la ciencia, la medicina se basa en la aplicación de los conocimientos previamente adquiridos para la resolución de los problemas que se puedan plantear. La principal diferencia entre la medicina y otras ciencias es la

vida humana. El objetivo de la sanidad es salvar vidas y mejorar el estado de salud del paciente. Es por ello que los simuladores cobran especial importancia, ya que un estudiante o un profesional, en la mayoría de las ocasiones, no puede poner en práctica los procedimientos a realizar en una situación real, ya que podría poner en riesgo la salud de una persona [16]. Se utilizan estos simuladores precisamente para entrenar esos procedimientos y habilidades, para representar todo tipo de escenarios que los profesionales médicos puedan encontrarse a lo largo de la vida profesional, y para que puedan adquirir esos conocimientos con éxito.

En relación con la enfermedad del trauma grave que se estudia en este Trabajo Fin de Grado, es importante tratar de manera rápida y eficiente al paciente traumatizado. Los primeros minutos y las primeras horas después de un accidente son esenciales para que el paciente viva y no sufra complicaciones ni consecuencias graves.
Consecuentemente, en este proyecto se ha desarrollado un simulador clínico virtual para el entrenamiento de las competencias y habilidades necesarias para hacer frente a los diferentes escenarios que puedan aparecer derivados de un traumatismo grave gracias a la ayuda de la tecnología y de los profesionales docentes [17].

1.3. Motivación y objetivos

La finalidad y motivación de este Trabajo Fin de Grado, como se ha mencionado anteriormente, es crear un simulador médico que ayude a los profesionales clínicos en su formación y mejore sus aptitudes para hacer frente a lesiones de trauma grave. Para ello, se tratarán de alcanzar los siguientes objetivos:

1. Diseñar una interfaz web en la que interactúen los profesionales clínicos en formación y los profesionales docentes encargados de la formación.

2. Facilitar la creación y personalización del escenario de la simulación por parte del clínico docente. De esta manera, será capaz de determinar las constantes vitales y la situación clínica de un paciente de trauma como si de una situación real se tratara.

3. Dar acceso al profesional en formación al caso clínico creado para que pueda llevar a cabo las acciones necesarias a través de la interacción con una interfaz web. Esta interfaz representará a un paciente virtual con las características expuestas por el docente.

4. Proporcionar un informe final con todas las acciones llevadas a cabo por el clínico en formación una vez terminado el entrenamiento.

5. Crear la interfaz por medio de una base de datos que albergue toda la información relacionada con los escenarios clínicos y con las acciones llevadas a cabo.

1.4. Planificación del proyecto

Para alcanzar los objetivos planteados en la sección anterior se han llevado a cabo las siguientes tareas, desarrolladas y detalladas en cada uno de los capítulos de este documento:

- El Capítulo 1 servirá como introducción a la enfermedad traumática grave y a los simuladores clínicos, protagonistas del desarrollo de este trabajo. Se tratará de poner en contexto el alcance que tiene esta enfermedad en la sociedad y la importancia del uso de simuladores en el ámbito de la medicina. De la misma manera que se detallarán los objetivos de este Trabajo Fin de Grado.
- El Capítulo 2 se centrará en el estudio e investigación de las lesiones traumáticas y su protocolo de actuación. Esto servirá de base para diseñar los requisitos del simulador interactivo a través de una aplicación web.
- En el Capítulo 3 se detallará el desarrollo de la aplicación web, en la que interactuarán los profesionales clínicos con distintos escenarios. Adicionalmente, se comentarán las herramientas utilizadas para su desarrollo.
- El Capítulo 4 explicará el proceso de diseño de la base de datos que registrará y guardará todos los datos de las simulaciones completas. Esto permitirá el seguimiento y la evaluación de los escenarios planteados, así como las acciones realizadas por los profesionales.
- En el Capítulo 5 servirá para conocer el despliegue de la aplicación y su conexión con la base de datos.
- Finalmente, en el Capítulo 6 se presentarán las conclusiones y las líneas futuras de este Trabajo Fin de Grado.

Capítulo 2

Traumatismos y protocolos de actuación

En este Capítulo se presentan los diferentes tipos de traumatismos y los protocolos de actuación que se utilizan para hacer frente a este tipo de lesiones. Estos protocolos son los que se van a integrar en un simulador clínico que permita a los profesionales adquirir las competencias y habilidades necesarias para hacer frente a este tipo de lesiones.

La enfermedad traumática grave es aquella en la que el paciente presenta lesiones de origen traumático, es decir, lesiones de los órganos o los tejidos causadas por una acción externa y que, además, pueden poner en riesgo la vida del paciente [18].
A continuación se presentan los diferentes traumatismos que puede sufrir un paciente traumatizado o politraumatizado.

- El **traumatismo craneoencefálico** es la causa más frecuente de daño cerebral tal y como se muestra en la Figura 2.1. El cráneo es el encargado de proteger al encéfalo, que junto con la médula espinal forman el Sistema Nervioso Central, caracterizado por ser el núcleo del procesamiento mental. Este traumatismo se clasifica en leve, moderado o grave según la Escala de Coma de Glasgow (GCS), siendo el traumatismo leve el más común con un 72 % de los casos, y los restantes tienen prácticamente la misma incidencia con un 16 % y un 12 % respectivamente [19]. La GCS sirve como medida del nivel de conciencia del paciente, siendo uno de los parámetros más importantes a tener en cuenta y dando una información concisa sobre el estado neurológico del paciente [5].
  La GCS observa y puntúa la respuesta verbal, ocular y motora, obteniendo una puntuación entre 3 y 15 puntos, siendo 3 el valor más grave del nivel de concienciencia y 15 el menos grave [20]. Se puntúa cada una de las respuestas de forma individual y, posteriormente, se suman las tres tal y como se puede apreciar en la Figura 2.2. Una vez realizado el examen al paciente y obtenida la puntuación en la escala, se podrá conocer el grado de gravedad del traumatismo.
  Debido a un traumatismo craneoencefálico, el cerebro puede sufrir daños. En primer lugar, por el impacto sufrido o por el movimiento de aceleración y desaceleración; y en segundo lugar por una lesión secundaria originada a partir de la primera y que puede ser desarrollada días después del incidente [21].

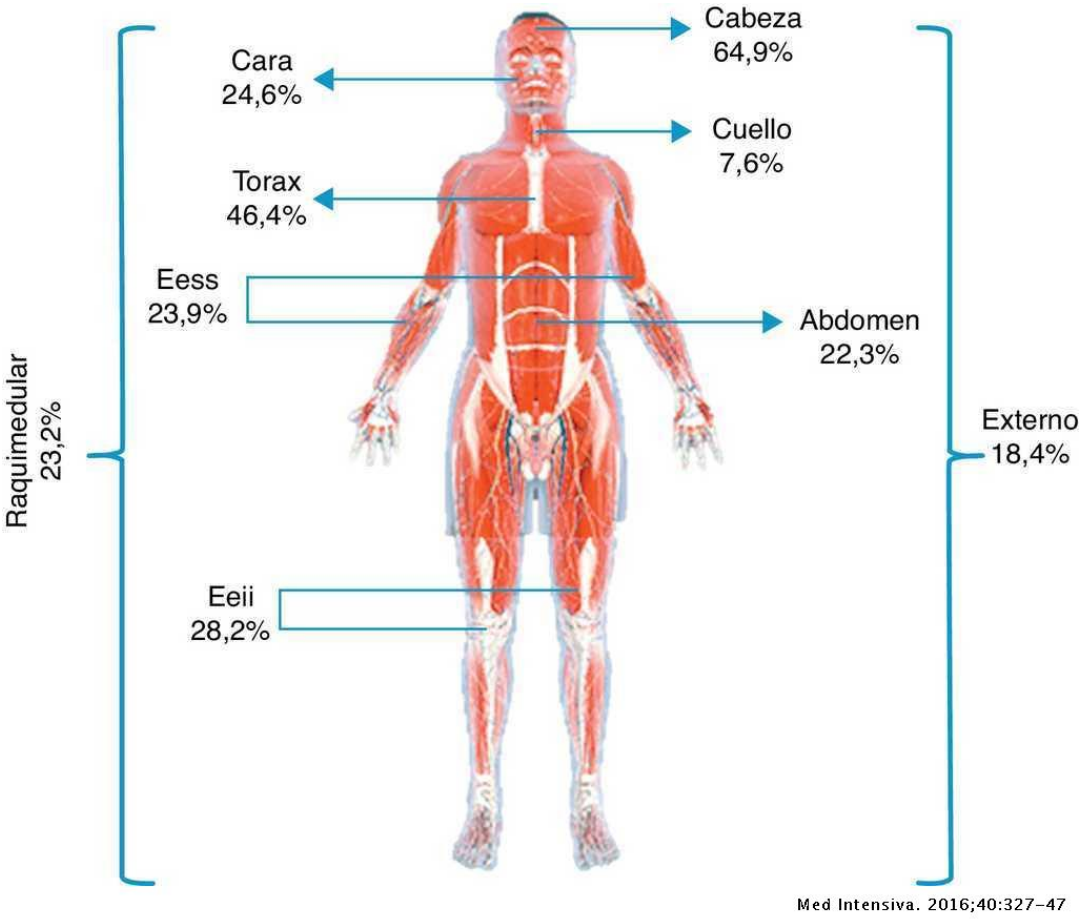

Figura 2.1: Localización de los diferentes traumatismos [4]

- Un **traumatismo abdominal** es una lesión que se origina en la zona del abdomen, y puede acarrear graves daños en el hígado, intestino, médula espinal o en los grandes vasos sanguíneos, y si no son tratadas rápida y adecuadamente, pueden resultar letales [22]. Estos traumatismos se pueden dividir en:
  - **Traumatismo cerrado:** se refiere principalmente a los traumatismos causados por un impacto directo o una disminución considerada de la aceleración. Los órganos más afectados suelen ser el bazo y el hígado.
  - **Traumatismo penetrante:** es aquel que se produce cuando un objeto perfora la piel. La transcendencia de la lesión dependerá de la profundidad de la herida y de la afectación de la cavidad abdominal, donde residen algunos órganos vitales o si tan solo se han dañado la grasa y los músculos que residen bajo la piel.
- Un **traumatismo torácico** es una lesión que se produce en el tórax y, al igual que el traumatismo abdominal, puede ser causado por el propio impacto o por heridas penetrantes. Estas lesiones pueden afectar a la pared ósea del tórax, la pleura, los pulmones, el diafragma, el corazón o los grandes vasos sanguíneos. La mayoría de los casos mortales se producen por complicaciones en la respiración y/o la circulación [23].

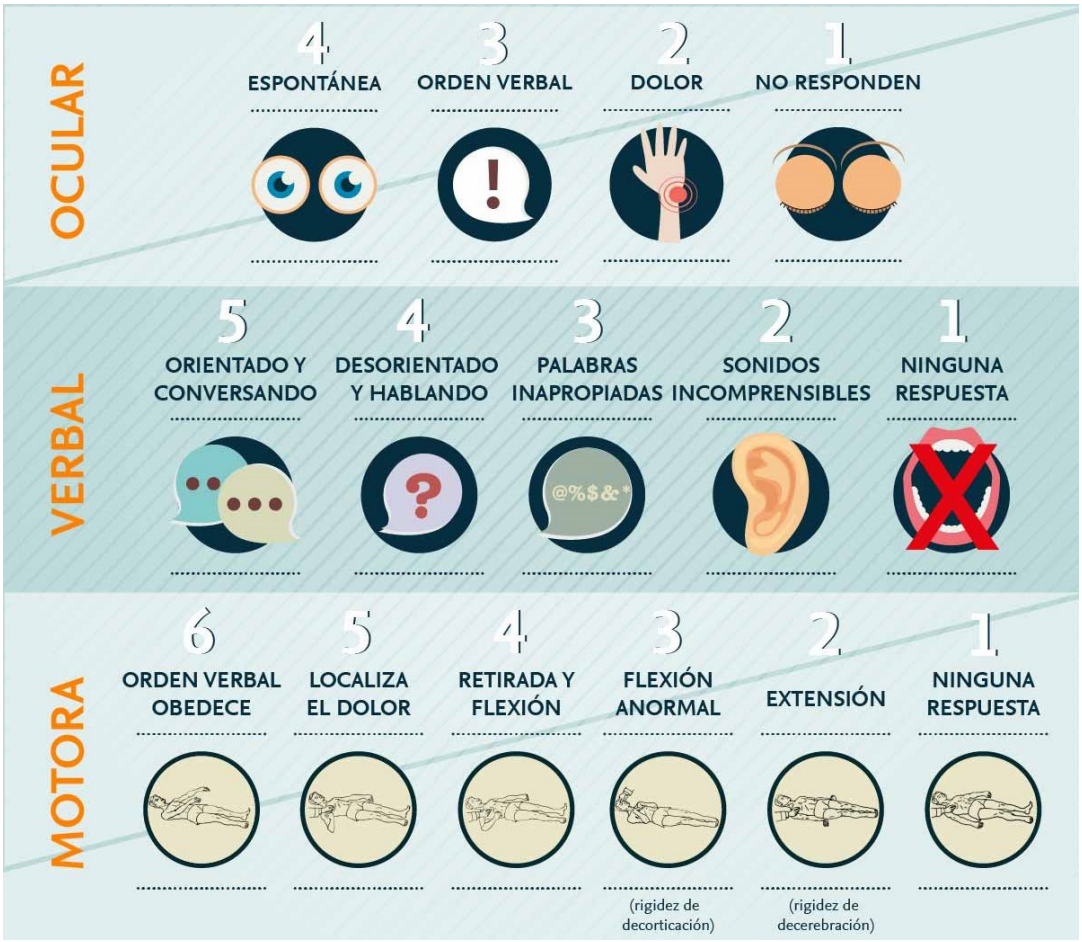

Figura 2.2: Tipos de respuesta y su puntuación en la GCS. [5]

- Una fractura de pelvis o un **traumatismo pélvico** afecta principalmente a la pelvis ósea, y por consecuencia también puede verse afectado el hueso sacro, el ilion, el isquion, el acetábulo o el pubis. Normalmente las personas mayores de 65 años son mayoritariamente las que padecen de estas lesiones, debidas en su mayoría a caídas accidentales [24].
  Las fracturas pélvicas dependen, al igual que todos los traumatismos, de la energía del impacto que sufren. Estos van desde las más leves con baja energía hasta las más graves asociadas a traumatismos de alta energía, que causan inestabilidad en el anillo pélvico, y pueden causar la muerte.
  Para disminuir la letalidad, se aplican protocolos basados en reducir el sangrado que origina este tipo de fracturas y estabilizar al paciente. A pesar de conseguir la estabilización, en la mayoría de las ocasiones, son muchas las secuelas que acarrea una fractura pélvica [25].

- En el momento que tiene lugar un traumatismo grave, **las extremidades** son la parte del cuerpo que mayor probabilidad tienen de sufrir un traumatismo, por detrás del traumatismo craneoencefálico y el torácico [9]. En este trabajo, sólo se consideran las lesiones de extremidades que pongan en riesgo la vida del paciente y/o que puedan conducir a la amputación del miembro.

La amputación se puede deber a que el hueso, los vasos sanguíneos, la piel o la parte nerviosa han sufrido grandes daños. Una máxima será la supervivencia del paciente, ante la conservación de un miembro. Para ello, lo primero que se debe evitar es el sangrado, frenando la hemorragia, por medio de diferentes técnicas y la inmovilización del miembro. Normalmente, se utilizan técnicas como los torniquetes, los vendajes hemostáticos y la presión manual [26].

2.1. Protocolo de actuación ante un traumatismo grave

Tras producirse un traumatismo grave es primordial que haya una coordinación extraordinaria entre todos los equipos que asistan a las víctimas. En España no existe un protocolo único de obligado cumplimiento en los centros hospitalarios cuando se atiende a un paciente traumatizado. Cada uno de estos centros han ido diseñando y aplicando diferentes protocolos que, en su mayoría, siguen un marco común basado en el curso estadounidense ATLS (Advanced Trauma Life Support) [27]. El ATLS es una guía de ayuda y entrenamiento orientada a médicos, para el tratamiento de pacientes que hayan sufrido un traumatismo, desarrollada por el Colegio Americano de Cirujanos. Surge en los años 80 en Estados Unidos con el objetivo de alcanzar un protocolo estandarizado ante pacientes traumáticos [28]. El ATLS se diferencia de otros protocolos en que muestra especial interés en realizar una primera evaluación exhaustiva pero a la vez rápida. De esta manera pueden detectar rápidamente cualquier lesión que pueda poner en peligro la vida del paciente realizando una exploración y evaluación primaria que les permite localizar o descartar lesiones que puedan acabar siendo mortales o que dejen daños irreparables. Todo ello en el menor tiempo posible, ya que los primeros momentos tras un accidente traumático son vitales para la evolución y recuperación del paciente [29]. Esta primera evaluación es conocida como evaluación 'ABCDE' y será detallada más adelante.

Llegados al momento en el que da comienzo la atención de un paciente traumatizado es importante llevar a cabo un proceso de triaje, que no es más que un proceso de clasificación que se emplea en el ámbito sanitario ante una emergencia con el objetivo de diferenciar a un paciente según su gravedad, su prioridad o necesidad de ser atendido.

2.2. Evaluación primaria

Un primer examen rápido y efectivo para medir el estado de un paciente que ha sufrido un traumatismo es la evaluación de diez segundos o también conocido como protocolo 'ABCDE', recomendado por el curso ATLS.
Su objetivo es la estabilización del paciente en el menor tiempo y la identificación y tratamiento de las lesiones que pongan en riesgo su vida.

2.2.1. A: control de la vía aérea

En primer lugar se debe comprobar y asegurar la permeabilidad de la vía aérea. Para ello se eleva el mentón y se realiza una inspección en busca de cuerpos extraños o alguna fractura que pueda obstruir la vía aérea. Esta búsqueda puede incluir la aspiración de sangre o secreciones.
Si el paciente responde al diálogo con normalidad, es un posible indicador de que la vía aérea no presente ninguna problemática. En caso contrario, y si el paciente muestra un nivel de consciencia menor de 9 en la escala de coma de Glasgow, será necesaria la intubación.

En el caso de que no se logre intubar al paciente, se establecerá una vía aérea alternativa de manera quirúrgica.
Por otro lado, hasta que no se demuestre lo contrario, se considerará que todos los pacientes tienen una lesión cervical, por lo que es obligatorio colocar un collarín para la estabilización cervical.

2.2.2. B: asegurar la oxigenación

Tras la comprobación de la permeabilidad de la vía aérea, es necesario asegurar una ventilación adecuada. Por ello se inspeccionan el tórax y el cuello valorando y examinando la presencia de respiración por medio de la frecuencia respiratoria y fijando una mascarilla con oxígeno.
También hay que descartar o tratar lesiones como el neumotórax o el hemotórax valorando la necesidad de dar soporte ventilatorio.

2.2.3. C: control hemodinámico

La hemorragia es el principal problema en las muertes evitables que se desarrollan a partir de un traumatismo. En primer lugar, es imprescindible identificar y tratar el shock, descartando o deteniendo tanto una hemorragia externa como interna.
Para identificar rápidamente si existe una hemorragia se debe medir el nivel de consciencia y el tiempo de relleno capilar, observar el color de la piel, el pulso y, paralelamente, se deben monitorizar las constantes vitales del paciente.
Ante una hemorragia externa, es preciso taponar, y si fuese necesario, realizar un torniquete. En caso de hemorragia interna, realizar una ecografía o un TAC (tomografía axial computarizada) para confirmar y tratarla.
Para finalizar, es posible que se tenga que reponer el volumen sanguíneo en el caso de que se produzca una importante pérdida de sangre, y siempre siguiendo los protocolos de transfusiones que tengan las entidades hospitalarias.

2.2.4. D: evaluación neurológica

Se procede a la evaluación neurológica comprobando el nivel de consciencia, el tamaño de las pupilas y la reacción del paciente.
Si se detectase un descenso del nivel de conciencia, se deberá hacer una reevaluación de las fases anteriores, descartando que existan lesiones que no hayan sido reconocidas hasta el momento.
Para estimar el nivel de conciencia se emplea la Escala de Coma de Glasgow, explicada al comienzo de la sección.

2.2.5. E: exposición del paciente

El paciente debe estar desnudo para facilitar su evaluación y las maniobras que tengan que realizarse, cubierto con mantas térmicas y en un ambiente cálido para prevenir la hipotermia. Los líquidos intravenosos que se inyecten tienen que ser calentados previamente, ya que mantener la temperatura óptima del paciente es vital. Si las lesiones lo permiten, se debe colocar una sonda vesical para medir la diuresis, indicadora de la cantidad de orina producida en un tiempo determinado por el paciente.

2.3. Evaluación Secundaria

Como se ha comentado anteriormente, los centros hospitalarios españoles no tienen un protocolo estandarizado ante la aparición una lesión traumática grave. Tanto es así, que cada centro sigue sus propios protocolos y tratamientos ante esta situación. Cabe destacar que en su mayoría, a grandes rasgos, son muy parecidos y están influenciados por el curso ATLS. Es por ello, que a continuación se comentan las características más generales e importantes que deben tener estos protocolos, haciendo distinción por el tipo de traumatismo que pueda ocurrir.

2.3.1. Traumatismo craneoencefálico

Al presentarse un traumatismo craneoencefálico, se procede a evaluar si el paciente se encuentra estable, y en caso negativo iniciar la evaluación 'ABCDE' para la estabilización de éste, y se avisa a la unidad de cuidados intensivos [30].
Si el paciente se encuentra estable se realizará un examen neurológico valorando un posible estado de amnesia. También se realizará una analítica para asegurarse que no se haya producido algún daño, y se valorará hacer un TAC en función de los síntomas que presente. Si el TAC resulta normal y el paciente presenta un buen estado será dado de alta. Por el contrario, si el estado aparentemente no es el adecuado, se seguirá su evolución en el propio hospital. Ante unos resultados que muestren alguna patología, se intervendrá quirúrgicamente.

2.3.2. Traumatismo abdominal

Tras reconocer que el paciente trasladado ha sufrido un traumatismo abdominal, se identifica si el traumatismo es cerrado o penetrante para actuar en consecuencia, siendo el primer grupo el más común debido a los accidentes de tráfico [31].
Tras el reconocimiento del traumatismo, se realiza una primera evaluación 'ABCDE' y se extrae una analítica, cuyos resultados reflejarán si el paciente se encuentra hemodinámicamente estable.

- **Trauma cerrado hemodinámicamente estable:** se procede a realizar una segunda evaluación centrada en la zona abdominal y un TAC abdominal para descartar encontrar órganos dañados o una víscera hueca. Ante el descarte de estas lesiones, se podrá dar de alta al paciente o mantenerle veinticuatro horas en observación para seguir su evolución. En caso contrario, y en función de la gravedad se procederá a su ingreso en planta o en la unidad de cuidados intensivos, valorando una posible cirugía de urgencia.

- **Trauma cerrado hemodinámicamente inestable:** cuando un paciente sufre inestabilidad hemodinámica, el objetivo es estabilizarle; ya sea por medio de analgésicos o cristaloides, como por medio de una transfusión sanguínea. Paralelamente se realizará un Eco-FAST u otra prueba de imagen para detectar la presencia de una hemorragia abdominal, o en caso necesario, realizar una laparotomía para inspeccionar los órganos abdominales.

- **Trauma abierto hemodinámicamente estable:** se debe realizar un examen de la herida. En caso de no ser penetrante, simplemente se observará la evolución del paciente. En caso contrario o de duda, se realizará un TAC o una ecografía abdominal, y en función de la gravedad se intervendrá quirúrgicamente, o se vigilará su evolución.

- **Trauma abierto hemodinámicamente inestable:** en este caso directamente el paciente será operado de urgencia.

2.3.3. Traumatismo torácico

El protocolo de actuación ante el trauma torácico dependerá de la energía del traumatismo [32]:

- **Baja energía:** en aquellos de baja energía se diferencia entre el traumatismo esternal y el traumatismo costal. Para el primero se realiza una radiografía para comprobar si existe una fractura. Si existe fractura, se estudiará realizar un ecocardiograma según dicten los resultados del electrocardiograma.
  Tras las pruebas pertinentes se valorará el ingreso del paciente y se observará su evolución.

En cuanto al trauma costal, una radiografía del tórax mostrará posibles facturas costales. En caso de encontrarse más de tres facturas, se valorará el ingreso del paciente. Sin embargo, si se encuentran una o dos fracturas, se examinará la zona costal afectada realizando un TAC, pudiendo ser dado de alta o ingresado según sus resultados.
Paralelamente es importante controlar el dolor, la saturación de oxígeno y la estabilidad del tórax.

- **Alta energía:** la evaluación 'ABCDE' precede a una segunda evaluación más precisa. En esta segunda evaluación se trata de identificar posibles lesiones que, en la mayoría de las ocasiones ponen en grave peligro al paciente.
  Estas lesiones pueden ser el neumotórax simple, el neumotórax a tensión o el neumotórax abierto; cuyo primer tratamiento es la colocación de un tubo torácico entre el quinto y el sexto espacio intercostal.
  Otra lesión podría ser el hemotórax masivo, que conlleva inestabilidad hemodinámica, y para la estabilización del paciente se coloca un tubo torácico en el quinto espacio intercostal.
  Además, podría aparecer un volet costal que requiere de oxigenoterapia y del ingreso en la UCI, controlando el dolor; ó también podría encontrarse un taponamiento cardiaco que sera tratado con una pericardiocentesis, toracotomía o esternotomía media.

2.3.4. Traumatismo pélvico

Ante un traumatismo pélvico y una posible fractura de pelvis, el protocolo a seguir será en primer lugar la evaluación 'ABCDE'. Y tras esta, conocer la estabilidad hemodinámica del paciente y fijar la pelvis, por ejemplo por medio de una faja o un cinturón pélvico. También se realizará una radiografía de tórax y pelvis.
Si existe inestabilidad hemodinámica, se realiza un Eco-FAST u otra prueba de imagen que permita encontrar alguna hemorragia en cuyo caso el paciente será intervenido quirúrgicamente. Si los resultados no muestran ninguna anomalía, se repetirá la prueba de imagen pasados unos minutos. En ambos casos si continúa la inestabilidad se optará por el protocolo de transfusión masiva y posterior fijación externa y de la pelvis.
Tras conseguir estabilizar al paciente, se le realizará un TAC con contraste para detectar algún sangrado adicional. Por último, se estudiará la estabilidad del anillo pélvico, cuyos resultados determinarán el procedimiento a realizar para fijar definitivamente el anillo pélvico. Esto puede ser a través de una cirugía a corto medio plazo, o si por el contrario, el anillo pélvico es inestable se procederá a una cirugía urgente para la fijación externa.

2.3.5. Traumatismo en extremidades

Al igual que en el tratamiento del trauma pélvico, se realiza en primer lugar una evaluación primaria 'ABCDE', y a partir de esta se diferencia entre pacientes hemodinámicamente estables y no estables:

- **Hemodinámicamente estable:** se realiza un examen secundario que determine las lesiones y su gravedad, y si es necesario se realizarán pruebas radiológicas. Entre las lesiones se pueden encontrar fracturas abiertas, luxaciones, lesiones vasculares, amputaciones o síndromes compartimentales.
  Después de la evaluación secundaria, valorando la gravedad, se determinarán los tratamientos a realizar. Los tratamientos podrán ser inmovilizaciones de los miembros, lavados, aplicación de frío, antibióticos, o incluso cirugía urgente.
  Finalmente se observará la evolución del paciente, y se estudiará su alta o su ingreso.

- **Hemodinámicamente inestable:** como se ha mencionado anteriormente, la misión es estabilizar al paciente. Esto se conseguirá controlando la hemorragia y reanimando al paciente a través de compresiones en la zona afectada, poniendo un torniquete o inmovilizando la zona, entre otros tratamientos.
  Es importante poner atención a las lesiones que puedan ser mortales, como la lesión neurovascular o el síndrome por aplastamiento.
  Tras la estabilización se procederá tal y como se describe en el punto anterior.

Capítulo 3

Desarrollo de la interfaz web

En este Capítulo se detalla el desarrollo de la aplicación web creada como simulador clínico para traumatismos en la que profesionales médicos puedan entrenar y mejorar sus habilidades. En las siguientes secciones se da una visión global de la aplicación y de las tecnologías y herramientas utilizadas para su desarrollo.

3.1. Descripción general de la aplicación

El simulador de trauma diseñado tiene como objetivo dar soporte a los profesionales clínicos del HULP en el entrenamiento de situaciones traumáticas graves. La aplicación software emula a un paciente virtual que presenta un caso específico de enfermedad traumática. Este caso es creado previamente por un profesional docente que establecerá los datos clínicos correspondientes que quiera recrear en el paciente virtual. Estos casos se asignarán a un estudiante que se pondrá al frente de la simulación para que pueda acceder en cualquier momento para iniciar su entrenamiento. La página principal de la aplicación es aquella en la que se lleva a cabo el entrenamiento y estará segmentada en tres componentes diferenciados:

1. Un **maniquí** que representa al paciente virtual y a los distintos tratamientos suministrados, adquiriendo así un grado de realismo comparable al de un maniquí físico, tal y como muestra la Figura 3.1

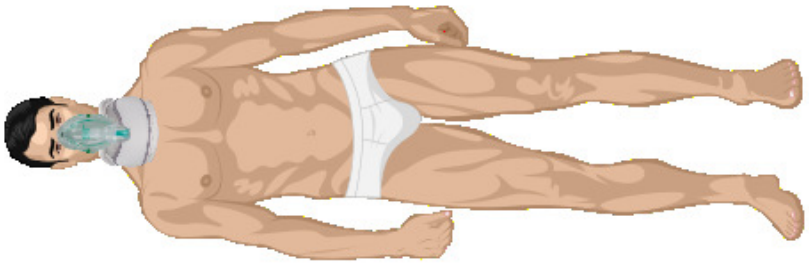

Figura 3.1: Maniquí con mascarilla de oxígeno y collarín cervical creado en el simulador.

2. Tres gráficas actualizadas constantemente en tiempo real que sirven de **monitor de constantes vitales**, según se muestra en la Figura 3.2. Estas tres gráficas representan la respuesta cardiaca, la respuesta respiratoria y la saturación de oxígeno del paciente en tiempo real y las distintas variaciones que se produzcan en ellas. Estarán acompañadas de otras constantes vitales como son la diuresis, la presión sistólica y la presión diastólica, tratando de hacer de esta parte un monitor de constantes como el que se encuentra en cualquier centro hospitalario.

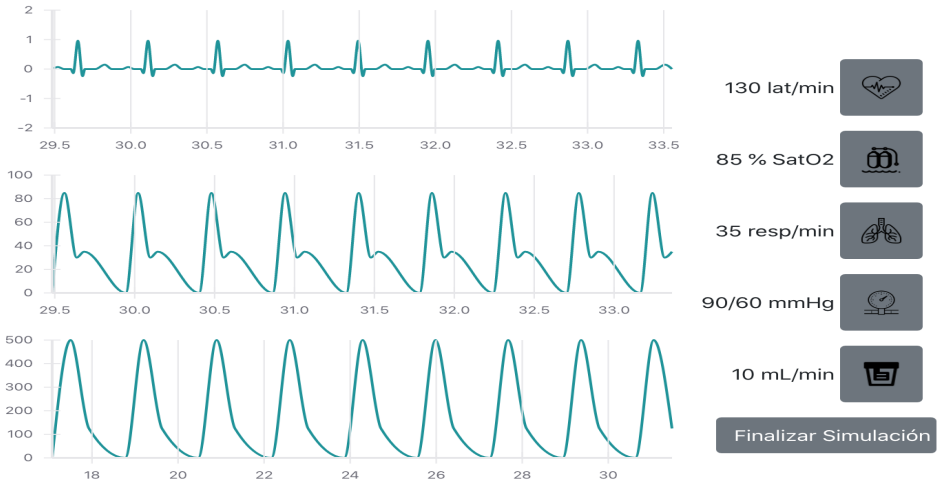

Figura 3.2: Monitor de constantes vitales

3. Un conjunto de **acciones**, representado en la Figura 3.3, que sirven para dotar al paciente de los cuidados requeridos, así como la posibilidad de realizar pruebas diagnósticas con el objetivo de recuperar y estabilizar al paciente. Al aplicar una acción concreta, ésta tendrá consecuencias en el resto de las partes de la interfaz, haciendo variar tanto las constantes vitales como el aspecto del maniquí, o la aparición de los resultados de una prueba realizada.

Figura 3.3: Algunas de las acciones que se aplican en el simulador.

Adicionalmente, a lo largo de la simulación se irán guardando todas las acciones llevadas a cabo por el estudiante generando un informe que se podrá obtener automáticamente tras la simulación y que se detalla en el Apéndice C.3.3. Este informe reflejará las acciones llevadas a cabo por el estudiante así como la variación de las constantes vitales del paciente virtual permitiendo al docente evaluar la simulación llevada a cabo. Además, este informe permite incluir comentarios de manera que el estudiante pueda tener una realimentación del trabajo llevado a cabo en la simulación.

En la aplicación web interactúan principalmente dos actores. Por un lado el instructor, que será un profesional médico, y por otro lado el estudiante, que será o un alumno de medicina o un clínico que quiera realizar la simulación para afianzar ciertas habilidades. En la siguiente sección se presentan los casos de uso en los que participan estos dos usuarios.

3.2. Descripción de los casos de uso

En el inicio del Trabajo Fin de Grado se plantearon una serie de casos de uso que han servido de guía en el desarrollo de la aplicación, detallados más abajo para poder entender las bases de esta. Los casos de uso representan a una serie de actores que interactúan en la aplicación por medio de un conjunto de escenarios [33]. En la Figura 3.4 se muestra gráficamente los casos de uso del simulador de trauma por medio de los diagramas UML (Lenguaje Unificado de Modelado). UML dispone de una notación gráfica específica para representar los casos de uso por medio de estos diagramas. Los casos de uso reflejan con mayor detalle la interacción de los actores, explicando los escenarios básicos o alternativos y los condiciones que se deben de dar al principio o al final de un escenario.

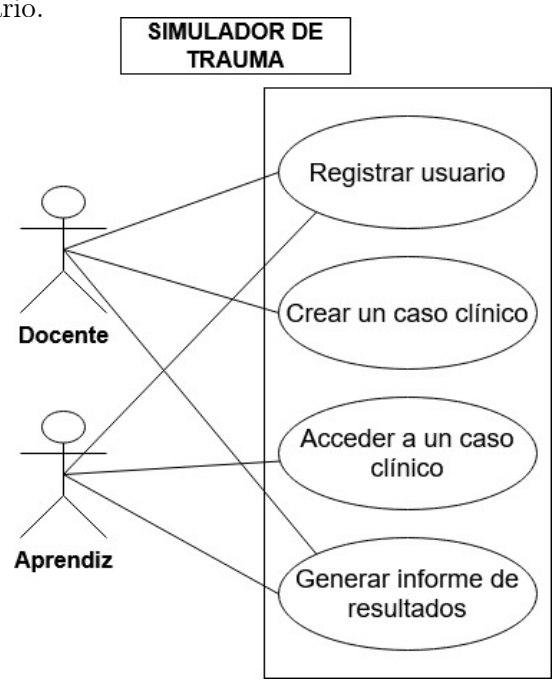

Figura 3.4: Diagrama de casos de uso.

3.2.1. Caso de uso 1

- **Caso de uso:** Registrar usuario.
- **Actores:** Instructor y estudiante.
- **Descripción:** Para poder utilizar el simulador de trauma es necesario registrarse previamente, introduciendo algunos datos personales.
- **Precondición:** Ninguna
- **Escenario normal:**
  1. Acceder a la página de registro de la página inicial de la web.
  2. Introducir los datos requeridos por el sistema y guardarlos.
  3. Si los datos son correctos, se mostrará una alerta informando que el registro ha tenido éxito.
- **Excepciones:**
  1. Si se produce algún error en el paso 3 al introducir los datos, se mostrará una alerta informando del error y se deberá volver al paso 2.
- **Postcondición:** El sistema mostrará la pagina de inicio para acceder a la aplicación.

3.2.2. Caso de uso 2

- **Caso de uso:** Crear un caso clínico específico.
- **Actores:** Instructor.
- **Descripción:** El instructor se encargará de preparar un caso clínico específico que emule a un paciente virtual con un traumatismo que tenga que ser tratado en un tiempo determinado.
- **Precondición:** El usuario debe estar registrado en el sistema y haber accedido a este.
- **Escenario normal:**
  1. El usuario accede a la página de creación de una simulación.
  2. El usuario introduce los datos específicos del paciente virtual, del caso clínico y le asigna a un estudiante el caso.
  3. Se mostrará una alerta informando que la creación se ha realizado correctamente.

- **Excepciones:**
  1. Si se produce algún error en el paso 3 al introducir los datos, se mostrará una alerta informando del error y se deberá volver al paso 2.
- **Postcondición:** El sistema mostrará el menú principal para crear una nueva simulación o acceder a las simulaciones creadas.

3.2.3. Caso de uso 3

- **Caso de uso:** Acceder a un caso clínico creado.
- **Actores:** Alumno.
- **Descripción:** El alumno tendrá acceso a la simulación de un caso clínico específico en un paciente virtual, sobre el que aplicar los tratamientos y conocimientos aprendidos en un tiempo limitado.
- **Precondición:** El usuario debe estar registrado en el sistema y haber accedido a este.
- **Escenario normal:**
  1. El usuario accede a la página de la simulación.
  2. El usuario inicia la simulación.
  3. El usuario aplica las acciones que se requieran hasta que el paciente virtual se estabilice o se agote el tiempo.
- **Postcondición:** El sistema mostrará el menú principal para acceder a las simulaciones y a sus resultados.

3.2.4. Caso de uso 4

- **Caso de uso:** Generar un informe de resultados de la simulación.
- **Actores:** Instructor y alumno.
- **Descripción:** Las distintas acciones llevadas a cabo por el alumno se recogerán para generar un informe de resultados; este informe se generará en paralelo a la simulación estando disponible una vez que termina la misma. Tanto el instructor como el alumno podrán acceder a este documento al finalizar la simulación para escribir comentarios o una evaluación por parte del instructor como para ver esta realimentación por parte del alumno.
- **Precondición:** Se debe estar realizando una simulación.

- **Escenario normal:**
  1. El alumno ejecuta una acción.
  2. El sistema guarda la información de la simulación en ese instante.
  3. Tras finalizar la simulación, el informe completo está disponible.
  4. El instructor realiza una realimentación en el informe.
  5. El alumno accede al informe para ver los comentarios.
- **Postcondición:** Ninguna.

3.3. Tecnologías utilizadas en el desarrollo del simulador

Para el desarrollo de los casos de uso que se conocían en la sección anterior se han utilizado diferentes tecnologías y herramientas que se detallan más adelante.
Este desarrollo ha estado guiado por el patrón MVC (Modelo-Vista-Controlador). El patrón MVC es un estilo de arquitectura de software cuya misión es separar la lógica de la aplicación o de control, de la lógica visual. Y para ello distingue tres componentes [34]:

- **Modelo:** representa y almacena los datos que maneja el sistema. En este trabajo el modelo se apoya en una base de datos MySQL. Toda la lógica del modelo de datos y la estructura de su base de datos requiere de un mayor detalle, y se especificará en el Capítulo 4.
- **Vista:** es el componente encargado de visualizar los datos del modelo. Presentan los datos de manera visual al usuario.
- **Controlador:** es el encargado de la interacción y comunicación del modelo y la vista. Adapta los datos para que encajen tanto en el modelo como en la vista.

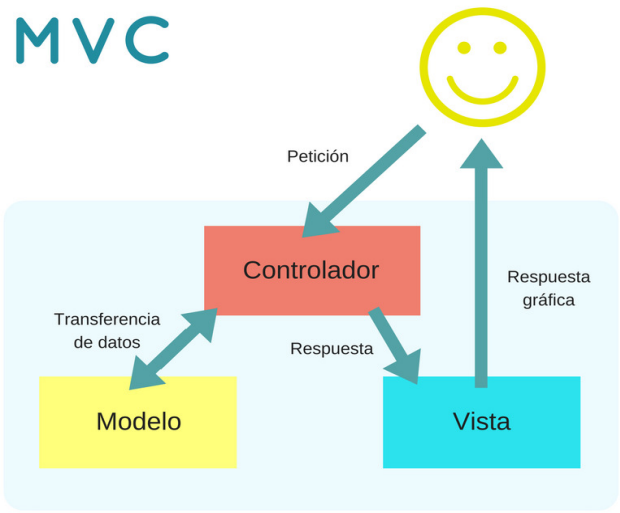

Figura 3.5: Patrón Modelo-Vista-Controlador[6].

En la Figura 3.5 se representa gráficamente el funcionamiento de este patrón. El cliente realiza una petición al servidor que por medio del controlador se encarga de manejarla. Esta petición podría incluir el almacenamiento de datos en el modelo o por el contrario podría requerir datos del modelo, de ahí que la flecha de la comunicación entre controlador y modelo sea bidireccional. Una vez manejada la petición se obtendrá una respuesta representada de manera gráfica en la vista que podrá ver el cliente en su terminal.
Para el desarrollo de la vista y el controlador del patrón MVC se ha utilizado el lenguaje JavaScript acompañado de distintos entornos comentados a continuación.

3.3.1. Javascript

JavaScript es un lenguaje de programación utilizado principalmente para la creación de web dinámicas [35]. JavaScript no necesita compilar sus programas para ejecutarlos. Es decir, como los programas se ejecutan del lado del cliente, son los propios navegadores los que se encargan de compilar el código. Es el complemento ideal a HTML y CSS en la creación de páginas web.

- **HTML:** es un lenguaje de marcado utilizado en el desarrollo de páginas web. A través de sus etiquetas o marcas de hipertexto se puede ordenar el contenido de una página web [36].

- **CSS:** las hojas de estilo CSS sirven para dar diseño a las etiquetas de marcado de HTML, estilizando las páginas web [37].

Además de desarrollar la parte visual de lado del cliente, también llamada frontend, JavaScript es capaz de diseñar la parte del servidor o del backend. Entre sus funciones también está la de realizar aplicaciones para dispositivos móviles, de escritorio o híbridas. Hay diversas librerías y entornos que se encargan de realizar todas estas funciones como React del lado del cliente, o el framework Express del lado del servidor, englobadas ambas en el entorno Node.js, que son las que se han utilizado en este Trabajo Fin de Grado.

3.3.2. React

React es una librería JavaScript encargada principalmente del desarrollo del frontend, de la parte visual de una aplicación. Este desarrollo se lleva a cabo de manera ordenada y escribiendo menos código que otras librerías gracias a su programación mediante componentes y a su DOM (Document Object Model) virtual [38].
Los componentes de React son los distintos fragmentos que forman una página web. Es decir, React construye una vista a partir de la unión de todos los componentes simplificados de la forma que el programador considere. Estos componentes reciben datos y devuelven lo que se desea mostrar. También disponen de un estado interno en el que almacenan esos datos, actualizando la vista cada vez que cambie el estado. Además un componente puede estar basado en uno o varios componentes, al igual que

se pueden reutilizar componentes, consiguiendo mayor eficiencia en el código.
El DOM es una representación de la interfaz gráfica de la aplicación que se ha creado. Por lo que ante una modificación de la interfaz, el DOM se actualizará y, como tiene una estructura en árbol, cada vez que se modifique un elemento, se actualizarán todos sus elementos hijos, lo que puede ralentizar la aplicación. React genera un DOM virtual cada vez que se produce un cambio en algún elemento de la interfaz. Antes de actualizar el DOM real, React compara su DOM virtual con el DOM real tratando de buscar el camino más optimo para realizar los cambios, haciendo más eficiente la actualización del DOM.
Se ha elegido React para el desarrollo del simulador debido a su facilidad en el diseño del frontend y su capacidad para dividir las páginas en componentes, pudiendo reutilizar estos. En la Figura 3.6 se muestran recuadrados los principales componentes creados para el simulador, reflejando la magnitud de esta biblioteca.

Figura 3.6: Componentes principales del simulador clínico: Barra de navegación, reloj, maniquí, acciones y monitor de constantes vitales.

Además, para que la aplicación sea agradable a la vista se han usado librerías como Bootstrap o Reactstrap, dando estilo a los diferentes componentes. Bootstrap es un framework CSS que incluye botones, barras de navegación o alertas entre otros muchos elementos que ya poseen un diseño atractivo para ser incluidos en una aplicación [39]. Por su parte, Reactstrap es la equivalencia de Bootstrap para la librería React.

3.3.3. Node.js

Node.js es un entorno JavaScript ejecutado del lado del servidor, asíncrono y orientado a eventos [40]. Esto significa que cuando el cliente realiza peticiones al servidor, Node trabaja con único hilo de ejecución en el que va acumulando eventos que no son más que peticiones que llegan por parte del cliente. Estos eventos se ejecutan de manera asíncrona, es decir, cada ejecución es independiente de las demás, permitiendo así que un solo servidor pueda mantener muchas conexiones, tal y como se representa en el esquema de la Figura 3.7.
Node puede beneficiarse de numerosas librerías y recursos que se instalan gracias al gestor de paquetes NPM (Node Package Manager), librerías como React comentada anteriormente para el diseño del frontend, o Express usada para el backend.

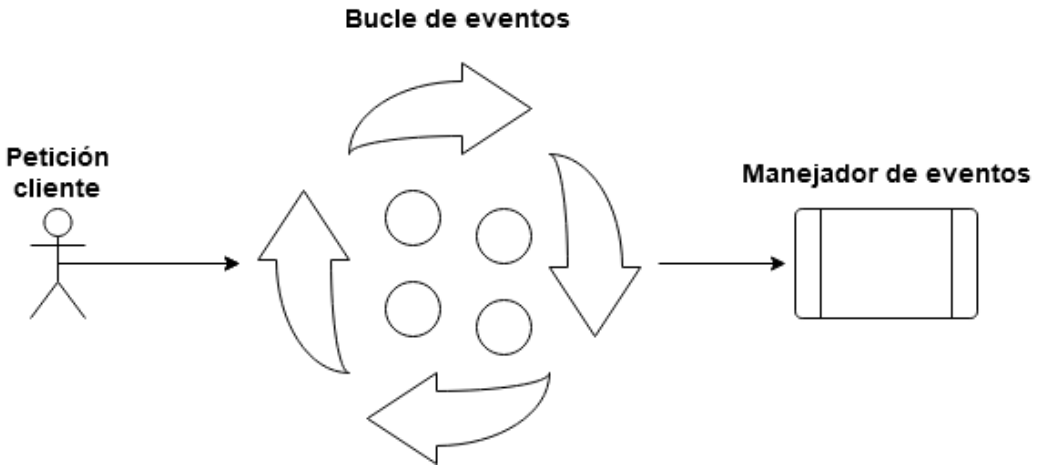

Figura 3.7: Esquema del proceso asíncrono y orientado a eventos de Node.js.

3.3.4. Express

Express es un framework de Node.js utilizado para implementar la arquitectura MVC [41]. Express puede generar automaticamente un esqueleto de directorios en los que se albergan los contenidos de la aplicación en los que destacan [42]:

- **Servidor:** carga el módulo *App.js*, que contiene la información de la aplicación desarrollada y el módulo *http* que es el propio servidor de la aplicación.
- **Controlador:** en Express, en la parte del controlador aparece el concepto de *middleware*. Un middleware es una función que recibe una entrada, una salida y se le es asignada una URL donde estén accesibles los dos atributos. Es por ello que en el simulador se ha utilizado esta parte para la creación de APIs(Aplication Programming Interface). Una API es un conjunto de funciones encargadas de llevar a cabo una o varias tareas y que puede ser usada por otro software.

- **Modelo:** en el modelo está la información de la base de datos y la conexión a ella. En el simulador se usará el paque Sequelize para la conexión y creación de las distintas tablas de la base de datos que serán detalladas en el Capítulo 4.

- **Vista:** el desarrollo de la parte visual en Express se realiza por medio del lenguaje de marcado HTML. Esto no supone un problema, pero utilizar React para llevar a cabo esta función dotará al simulador de mayor eficiencia, como ya se ha explicado anteriormente.

Así que Express se ha utilizado para la creación de las APIs encargadas de suministrar los datos provenientes del modelo a su correspondiente vista o del tratamiento de los datos que se devuelven desde las vistas a través de peticiones HTTP.

3.4. Frontend: Diseño y estructura.

En este apartado se explica cómo está formada la parte visual con la ayuda de los flujos de navegación de la aplicación que están representados en la Figura 3.8. Instalar Node.js es un requisito imprescindible para crear una aplicación de React. Usando el gestor de paquetes NPM se procede a la instalación del módulo 'create-react-app', encargado de la creación de una aplicación en React. Tras la ejecución de este módulo aparecen ciertos archivos y carpetas que formarán el esqueleto de la aplicación:

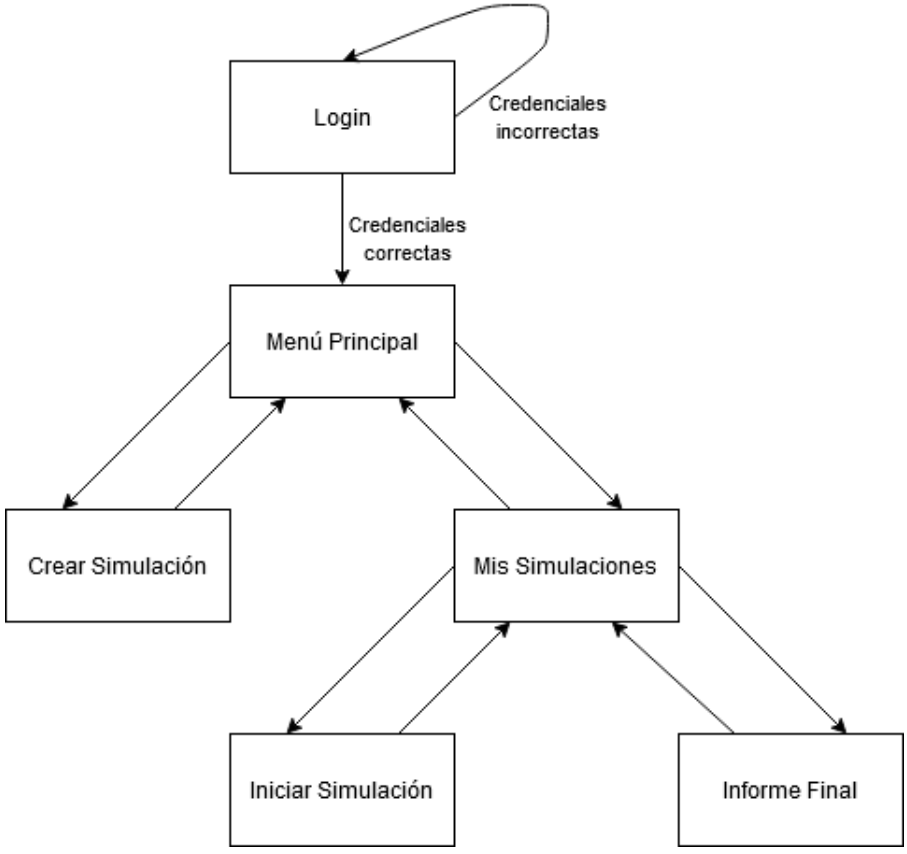

Figura 3.8: Flujo de navegación de la aplicación.

- **node_modules:** directorio en el que se ubican todas las librerías que se desean instalar, así como todas las dependencias instaladas por defecto para que el simulador funcione correctamente.
- **public:** dentro de esta carpeta se encuentra el archivo más importante de toda la aplicación creada con React, el archivo index.html. En este archivo se irán renderizando todos los componentes del simulador. Adicionalmente se incluyen archivos a nivel local que podrán ser utilizados por los componentes.
- **src:** directorio en el se albergan todos los componentes diseñados que darán lugar a las diferentes vistas del simulador.
- **README.md:** breve manual de usuario con información sobre la aplicación.
- **package.json:** archivo que tiene todo proyecto de Node de que contiene toda la información de la aplicación y es controlado por el gestor de paquetes NPM. Cabe destacar que guarda el nombre de todas las dependencias instaladas, para que cuando se configure la aplicación se proceda a su instalación.
- **yarn.lock:** archivo similar al anterior con la diferencia que tan solo alberga la información de los módulos instalados en la aplicación.

Figura 3.9: Estructura del frontend.

En la Figura 3.9 se puede apreciar el esqueleto del frontend del simulador clínico y todos los componentes creados en la carpeta src que dan lugar a las diferentes vistas que aparecen en el diagrama de flujos de la Figura 3.8.
Y para finalizar con la parte visual del proyecto, hay que resaltar el uso del entorno ***I18next***. Se trata de un entorno de internacionalización escrito en JavaScript que sirve para traducir el contenido de una aplicación a un determinado idioma por medio de un fichero en el que se han almacenado las traducciones correspondientes a cada idioma.

3.5. Backend: Diseño y estructura

El backend forma la parte lógica del simulador, servirá y recibirá los datos al frontend, en este caso a la aplicación React creada. Para crear un servidor basado en Express, es necesario crear un *package.json* a través del gestor de paquetes NPM, con la información de la aplicación y las dependecias instaladas tal y como muestra la Figura 3.10.

backend > {} package.json > ...

```
{
  "name": "backend",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "dev": "nodemon src/App.js",
    "migrate": "./node_modules/.bin/sequelize db:migrate --url mysql://root:luis110797@mysql:3306/trauma_simulator",
    "seed": "./node_modules/.bin/sequelize db:seed:all --url mysql://root:luis110797@mysql:3306/trauma_simulator"
  },
  "author": "lcastaneda",
  "license": "ISC",
  "dependencies": {
    "express": "^4.17.1",
    "mysql2": "^2.1.0",
    "sequelize": "^5.21.5",
    "sequelize-cli": "^5.5.1"
  },
  "devDependencies": {
    "nodemon": "^2.0.3"
  }
}
```

Figura 3.10: package.json

Express permite, instalando su paquete *express-generator*, crear un esqueleto para la creación del servidor, tal y como se detallaba en la explicación que se ha dado sobre el framework Express. Sin embargo, como se utilizará React para desarrollar la parte visual, se ha personalizado el propio esqueleto específicamente para el simulador y la construcción de sus APIs quedando de la siguiente forma:

- **node_modules:** directorio en el que se ubican todos las librerías instaladas, como es el caso de Express.
- **model:** aquí está albergada la parte del modelo de los datos del simulador, y donde tiene lugar la conexión con la base de datos.
- **routes:** directorio donde se escriben todas las rutas de las APIs creadas, para que puedan ser llamadas desde el frontend.
- **controlers:** directorio donde se ubican todos los archivos que contienen la parte lógica de las APIs, es decir, donde se definen las funciones que se llevarán a cabo.
- **App.js:** es el archivo más importante. En él se define y se inicializa la aplicación Express que dará lugar a todas las APIs. En él se especifican algunas de las configuraciones de la aplicación, como el puerto TCP/IP por donde se tendrá acceso a las diferentes APIs.

Capítulo 4

Diseño y modelo de la base de datos relacional

Durante el desarrollo del simulador ha surgido la necesidad de almacenar los datos utilizados y generados por la aplicación. En primer lugar, se almacenan los datos pertenecientes a los usuarios que interactuarán con la aplicación, así como su rol dentro de la aplicación, de manera que el sistema pueda controlar los accesos a los diferentes recursos. En segundo lugar también es necesario guardar los registros pertenecientes a una simulación, en la que se albergan los datos médicos del paciente virtual y del caso clínico que se creará. Por estas razones se ha decidido utilizar un software que se encargue de administrar y almacenar toda la información relacionada con el simulador clínico en una base de datos. Entre estos gestores de bases de datos destaca MySQL, utilizado en este trabajo para llevar a cabo esta tarea.

4.1. MySQL

MySQL es un sistema de administración de bases de datos relacionales de código abierto a través de un modelo cliente-servidor que permite crear y administrar bases de datos basadas en un modelo relacional[43].

- Una **base de datos** es un conjunto de datos almacenados y organizados de manera estructurada. El término relacional se refiere a que los datos se almacenan en diferentes tablas interconectadas entre sí. Es decir, MySQL se encarga de crear una base de datos cuyas tablas guardan relación unas con otras.

- Un sistema de **código abierto** quiere decir que es accesible y gratuito para todo aquel que desee utilizarlo e incluso modificarlo para adaptarlo a sus necesidades. Gracias a estas características MySQL está en constante desarrollo ofreciendo numerosas actualizaciones que mejoran el servicio.

- MySQL utiliza una arquitectura **cliente-servidor**. Los terminales donde se instala el software de MySQL son los clientes y cuando estos necesitan acceder a los datos almacenados en las tablas de la base de datos se comunican con el

servidor. Así cada cliente se comunicará con el servidor realizando una solicitud y el servidor le enviará una respuesta con la información requerida, tal y como se aprecia en el esquema de la Figura 4.1. Estas solicitudes se llevan a cabo por medio del lenguaje SQL (Structured Query Language) que se detalla a continuación [44].

Figura 4.1: Arquitectura cliente-servidor en MySQL.

4.2. SQL

A menudo es fácil confundir MySQL y SQL, pero no son iguales. MySQL, como ya se ha dicho, es un software encargado de crear una base de datos relacional a través de un modelo cliente-servidor. Mientras que SQL es el lenguaje que utilizan tanto cliente como servidor para su comunicación. SQL es capaz de manejar las peticiones del cliente indicando qué debe hacer el servidor, limitándose a llevar a cabo una serie de operaciones, entre las que destacan[45]:

- El cliente puede **consultar datos** solicitando cualquier información de la base de datos.
- El cliente es capaz de **modificar datos** agregando, eliminando, cambiando u ordenando la información que se encuentra en la base de datos.
- Se puede **definir** la naturaleza de los propios datos. Es decir, indicar las propiedades de los datos que se encuentran en una tabla de la base de datos e indicar las relaciones entre las diferentes tablas.
- Es posible **censurar la información** con el objetivo de proteger los datos, indicando quien tiene acceso a la información almacenada en la base de datos.

A continuación se detallan las principales instrucciones SQL encargadas de llevar a cabo las operaciones de la lista anterior:

- **SELECT:** sirve para consultar datos.
- **INSERT:** instrucción para insertar datos en una tabla.

- **UPDATE:** instrucción para actualizar o modificar datos existentes que forman parte de una tabla de la base datos.
- **DELETE:** se utiliza para borrar datos.
- **DISTINCT:** elimina las consultas de datos duplicadas, en el caso de que existan duplicados.
- **WHERE:** filtra los datos que queremos consultar imponiendo ciertas condiciones.
- **AND y OR:** operadores lógicos utilizados para incluir más de una condición en una consulta.
- **ORDER BY:** se utiliza para ordenar los datos obtenidos de una consulta.

4.3. Sequelize

Además de crear una base de datos, es importante que esta pueda sincronizarse con la parte del backend detallada en el Capítulo 3 donde el controlador será el encargado de manipular la información proveniente de la base de datos. Es decir, las tablas creadas en la base de datos deben ser compatibles con el desarrollo del backend. Esto se consigue gracias a un ORM (Object-Relational Mapping), que es un modelo de programación orientado a objetos que permite mapear las tablas de una base de datos relacional a través de una entidad, es decir transforma los datos almacenados en objetos que puedan ser utilizados por la aplicación o viceversa [46]. Además permite dejar a un lado las consultas y gestiones manuales a partir del lenguaje SQL, pasando esto a realizarse por parte del ORM con su lenguaje propio, obteniendo un desarrollo cómodo y efectivo de las aplicaciones.

Sequelize es un ORM para Node.js que permite ejecutar funciones JavaScript para interactuar y manipular bases de datos SQL, entre ellas MySQL, sin necesidad de introducir consultas SQL, ya que vienen implícitas en estas funciones javascript. Sequelize se puede instalar por medio del gestor de paquetes NPM y se adapta perfectamente al entorno Express utilizado para el desarrollo del backend. Aunque Sequelize aporte numerosas funciones que evitan consultas SQL manuales, es posible también introducir estas en caso de que se quiera realizar una consulta compleja o personalizada que se escape de los límites de las funciones Sequelize. Además Sequelize ofrece diseñar el esqueleto de las tablas de la base de datos para que sean creadas durante la instalación de la aplicación, es lo que se conoce como migración. También existen las semillas, que se utilizan para introducir información en las tablas creadas durante la migración.

4.4. Diseño del modelo del simulador clínico

Para diseñar la base de datos del simulador clínico se ha utilizado MySQL como gestor de la base de datos y Sequelize como ORM ejecutado del lado del backend en el entorno Express. A continuación se detalla el proceso que se ha seguido para crear la base de datos completa que contiene los datos del simulador, así como los diferentes modelos utilizados para la creación de las tablas.

1. Para **crear una base de datos** por medio de Sequelize se ha escrito un archivo de configuración como el de la Figura 4.2 en el que se indique el nombre de la base de datos que se creará, el gestor utilizado, que en este caso se trata de MySQL y las credenciales para poder acceder al gestor.

```
var Sequelize = require('sequelize');

const sequelize = new Sequelize(
  'trauma_simulator',
  'root',
  'luis110797',
  {
    host: 'mysql',
    dialect: 'mysql',
    port: '3306'
  }
);

module.exports = sequelize;
```

Figura 4.2: Archivo de configuración Sequelize de la base de datos MySQL

2. **S**e han diseñado los modelos que definen las entidades a partir de las cuales se crean las tablas de la base de datos. En cada una de estas entidades se incluyen los tipos de datos que se requieren introducir en cada campo, esto se muestra en las Tablas 4.1, 4.2, 4.3 y 4.4, que representan las diferentes tablas de la base de datos del simulador clínico.

Entidad		**Atributo**				
Nombre	Descripción	Nombre	Descripción	Tipo	PK	Obligatorio
Trainer	Datos personales del instructor	**TrainerId**	Identificador del instructor	INTEGER	Sí	Sí
		Name	Nombre del instructor	STRING	No	Sí
		Surname	Apellidos del instructor	STRING	No	Sí
		Email	Email del instructor	STRING	No	Sí
		Password	Contraseña del instructor	STRING	No	Sí
		RoleId	Identificador del rol del instructor	INTEGER	No	Sí
		Workplace	Lugar de trabajo del instructor	STRING	No	Sí

Tabla 4.1: Modelo de la tabla del instructor.

Entidad		Atributo				
Nombre	Descripción	Nombre	Descripción	Tipo	PK	Obligatorio
Trainee	Datos personales del instructor	**TraineeId**	Identificador del estudiante	INTEGER	Sí	Sí
		Name	Nombre del estudiante	STRING	No	Sí
		Surname	Apellidos del estudiante	STRING	No	Sí
		Email	Email del estudiante	STRING	No	Sí
		Password	Contraseña del estudiante	STRING	No	Sí
		RoleId	Identificador del rol del estudiante	INTEGER	No	Sí
		Workplace	Lugar de trabajo del estudiante	STRING	No	Sí

Tabla 4.2: Modelo de la tabla del estudiante.

Entidad		Atributo				
Nombre	Descripción	Nombre	Descripción	Tipo	PK	Obligatorio
Roles	Información sobre el puesto que ocupa el usuario de la aplicación	**RoleId**	Identificador del rol	INTEGER	Sí	Sí
		Role	Nombre del puesto o situación laboral del usuario	STRING	No	Sí

Tabla 4.3: Modelo de la tabla de los roles.

3. Como se ha comentado, las tablas de la base de datos guardan relación entre sí. Estas relaciones se establecen a través de dos claves. La PK (Clave Primaria) que es uno de los atributos incluidos en una tabla de la base de datos. Este atributo también estará incluido en la tabla con la que se establezca la relación, y en esta segunda tabla aparecerá como FK(Clave Externa). Es decir, cuando dos tablas se quieren relacionar, la clave primaria de la primera aparecerá como clave externa en la segunda y/ó viceversa, dependiendo si se relacionan en un único sentido o en ambos sentidos.
   Las relaciones que guardan entre sí las tablas de la base de datos del simulador se pueden ver representadas en la Figura 4.3.

Entidad		Atributo				
Nombre	Descripción	Nombre	Descripción	Tipo	PK	Obligatorio
Simulation	Datos del caso clínico especificado por el instructor.	**SimulationId**	Identificador de la simulación.	INTEGER	Sí	Sí
		TrainerId	Identificador del instructor.	INTEGER	No	Sí
		TraineeId	Identificador del estudiante.	INTEGER	No	Sí
		Sex	Sexo del paciente (0-Masculino 1-Femenino)	INTEGER	No	Sí
		Age	Edad del paciente.	INTEGER	No	Sí
		Weight	Peso del paciente.	DOUBLE	No	Sí
		PartBody	Parte del cuerpo del paciente afectada.	STRING	No	Sí
		MentalStatus	Estado mental del paciente.	STRING	No	Sí
		BloodLoss	Sangre perdida por el paciente.	DOUBLE	No	Sí
		SistolicPressure	Presión arterial sistólica del paciente.	DOUBLE	No	Sí
		DiastolicPressure	Presión arterial del paciente.	DOUBLE	No	Sí
		HeartRate	Frecuencia cardiaca del paciente.	DOUBLE	No	Sí
		BreathingRate	Frecuencia respiratoria del paciente.	DOUBLE	No	Sí
		UrineOutput	Flujo urinario del paciente.	DOUBLE	No	Sí
		Saturation	Saturación sanguínea del paciente.	DOUBLE	No	Sí
		Temperature	Temperatura corporal del paciente.	DOUBLE	No	Sí
		Time	Tiempo límite para realizar la simulación.	INTEGER	No	Sí

Tabla 4.4: Modelo de la tabla de la simulación.

4. La definición del modelo y las relaciones detalladas en los puntos anteriores hacen que estas entidades puedan ser utilizadas como objetos JavaScript en el proyecto Express que se ha desarrollado. Para que se creen las tablas en la base de datos MySQL se necesita escribir un fichero de migraciones que contendrá de nuevo la definición de esos modelos y relaciones. Tras ejecutar esas migraciones se crearán las tablas en la base de datos.

5. Si se desea rellenar las tablas creadas a partir de las diferentes migraciones, se puede realizar de dos modos. Por un lado, a través del gestor MySQL y por medio de instrucciones SQL se irán introduciendo los datos deseados. Por otro lado, se puede crear un fichero de semillas, similar al de migraciones, ambos utilizados en este trabajo, que se encargue de rellenar las tablas como se desee.

Tras seguir la secuencia anterior se ha obtenido una base de datos MySQL que cuenta con la estructura que representa la Figura 4.3.

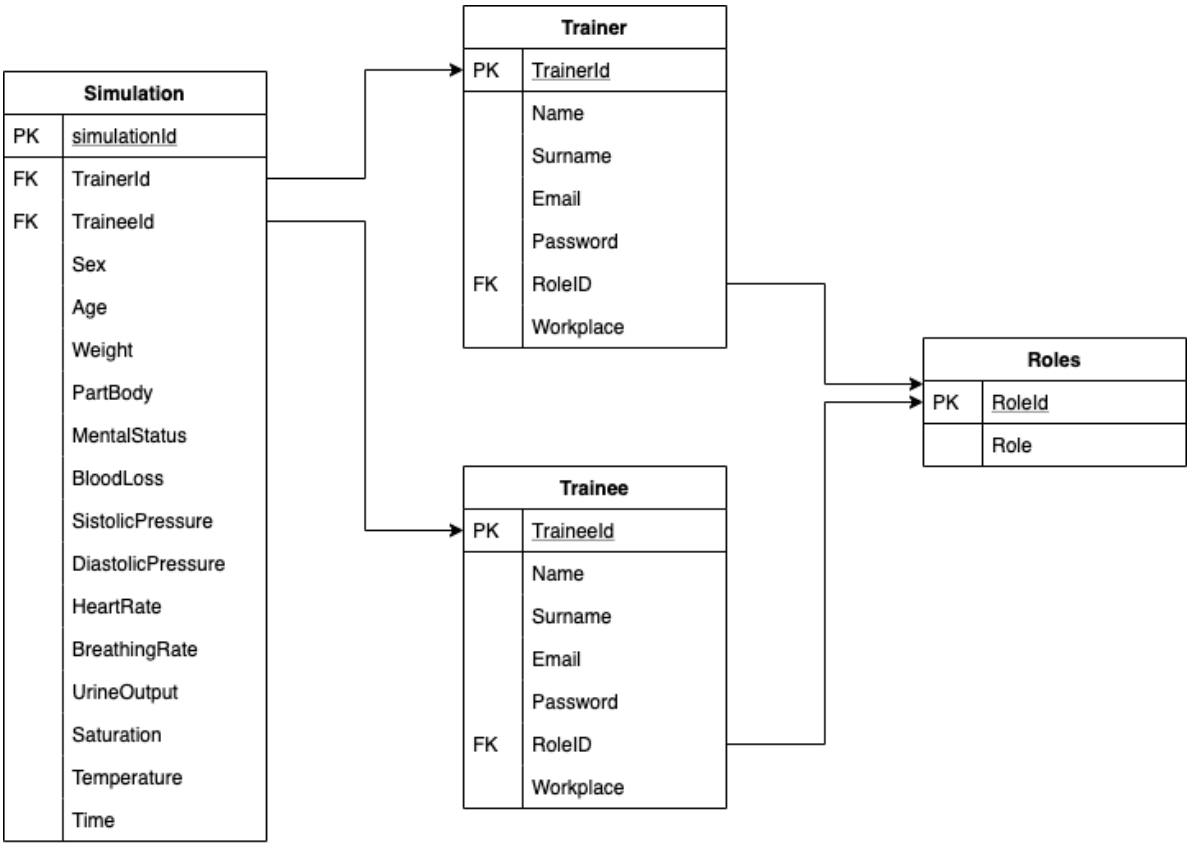

Figura 4.3: Modelo relacional de la base de datos del simulador clínico.

Capítulo 5

Despliegue del simulador clínico en Docker

Tras haber desarrollado la parte visual y la parte del servidor del simulador, se procede al despliegue de la aplicación al completo para que pueda ser utilizada por cualquier profesional médico y en cualquier dispositivo. Así pues, se va a utilizar Docker, a través del cual se ha llevado a cabo el despliegue de la aplicación.

5.1. Docker

Docker es una plataforma software de virtualización que permite crear contenedores ligeros y portables para empaquetar una aplicación acompañada de sus dependencias y librerías necesarias para su ejecución [47]. Docker ofrece la posibilidad de ejecutar una aplicación en cualquier entorno, independientemente del sistema operativo del que disponga la máquina principal. Analizando la definición que se acaba de dar, resaltan dos términos como son la virtualización y los contenedores:

- La **virtualización** se refiere a la creación de una versión virtual de un recurso tecnológico creando una capa de abstracción entre el hardware de la máquina física y el sistema operativo de la máquina virtual. Esta capa es conocida como hipervisor. El hipervisor oculta las características del hardware físico para instalar el sistema operativo virtual en su capa superior, como muestra la imagen de la izquierda de la Figura 5.1. La creación de estas máquinas virtuales suelen consumir demasiados recursos de la máquina física, ya que cada máquina virtual aisla un sistema operativo instalado en la máquina física. Sin embargo, Docker es una plataforma software de virtualización a nivel de sistema operativo. Es decir, Docker trabaja directamente con el sistema operativo de la máquina física en la que se ejecuta el contenedor. Por lo tanto, Docker además de abstraer la parte física de la máquina, consigue también abstraer el sistema operativo, trabajando directamente sobre este y repartiendo sus recursos a los contenedores. De esta manera los contenedores compartirán los recursos del sistema operativo físico aislando las aplicaciones, como se refleja en la Figura 5.1. Con ello se recupera gran parte de los recursos que se consumen en una virtualización gracias a un proceso más eficiente y ligero, tal y como decía la definición de Docker.

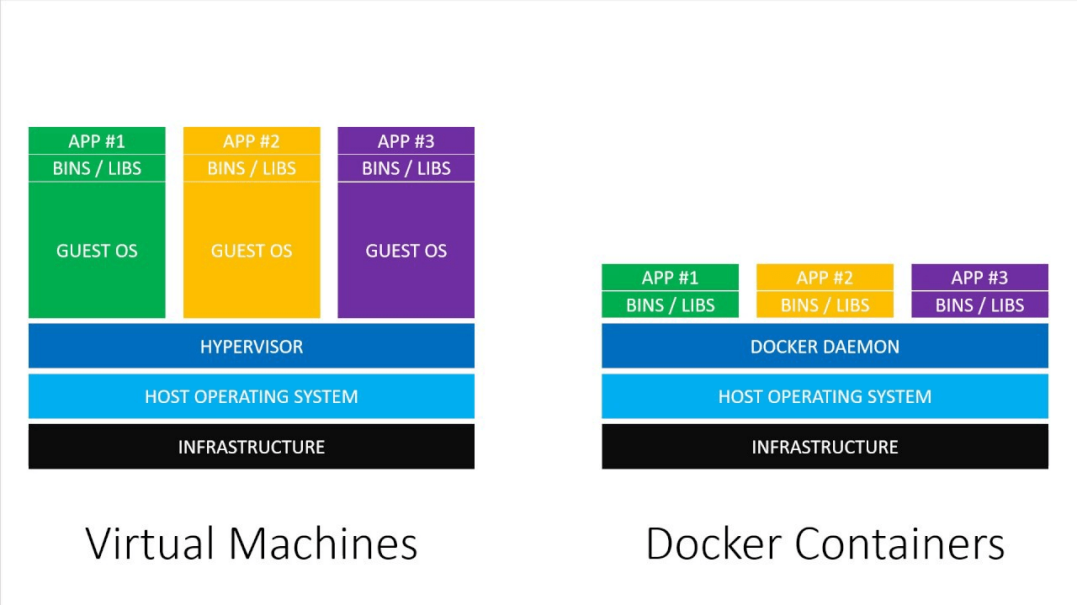

Figura 5.1: Diferencias entre virtualizaciones[7]

- El **contenedor** Docker tiene la misma misión que tiene un simple contenedor mercantil. Un contenedor sirve para almacenar objetos y que sean transportados de un lugar a otro. Pues bien, un contenedor Docker almacena todas las librerías y dependencias necesarias para la ejecución de una determinada aplicación, y gracias a almacenar estas, es posible ejecutar esa aplicación en cualquier entorno. Por lo tanto, se confirma que un contenedor Docker es portable.

Los contenedores siempre están ligados a imágenes Docker. Una imagen se utiliza para crear un contenedor. En ella se encuentra el código y las configuraciones del servicio que se ejecutará en el contenedor. Las imágenes son inmutables y están formadas por capas de sólo lectura [48]. Esto significa que tras crear una imagen, esta nunca se modifica. Al contrario que los contenedores que, aunque estén creados a partir de una imagen, sí son mutables. Cuando una imagen crea un contenedor, se añade una capa de lectura-escritura a las capas de lectura de la imagen. Así todos los cambios que se produzcan en el contenedor se almacenarán en esta capa y se recuperarán en cada inicio del contenedor. La Figura 5.2 ayuda a comprender lo anterior y muestra que se pueden crear múltiples contenedores idénticos a partir de una imagen. Estos contenedores dejaran de ser iguales si se producen modificaciones una vez arrancado el contenedor.

Existen imágenes ya creadas que se pueden utilizar y consultar en el registro público Docker Hub. Si se desea obtener una imagen personalizada se debe escribir el archivo de configuración Dockerfile a través del cual se crean las imágenes. Un Dockerfile es un archivo de texto plano que actúa como un manual de instrucciones que se debe seguir para crear una imagen, que será transformada posteriormente en un contenedor Docker.

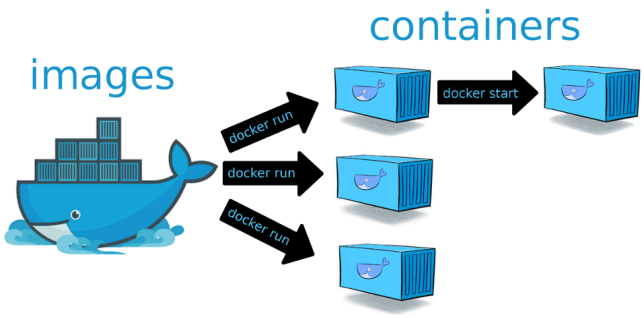

Figura 5.2: Creación de diferentes contenedores a partir de una imagen[8].

Normalmente una imagen está formada por una imagen base existente que instala un sistema operativo para el contenedor. Y a partir de esta imagen se darán las instrucciones necesarias para personalizar la imagen que se creará.
Además de ejecutar los contenedores de manera individual, como se muestra en la Figura 5.2, también se pueden ejecutar varios contenedores en conjunto a través de docker-compose. Compose es una herramienta que sirve para arrancar imágenes Docker de múltiples contenedores. Los contenedores Docker suelen dar un servicio individualizado, así para desplegar una aplicación en Docker será necesario ejecutar varios servicios como pueden ser un servidor o una base de datos relacionados entre sí, y esto se consigue gracias a ejecutar conjuntamente estos servicios. Compose utiliza el archivo *docker-compose.yml* para configurar todos los servicios o contenedores que se quieran ejecutar.

5.2. Despliegue del simulador clínico

Como se ha explicado en los capítulos anteriores, el simulador clínico está formado por el frontend y el backend. Éste último además soporta una base de datos MySQL. Estas tres partes, frontend, backend y la base de datos, se han diseñado para que su funcionamiento sea conjunto, por lo que se ha decido llevar a cabo el despliegue del simulador utilizando la tecnología de Docker, ideal para desplegar aplicaciones web gracias a su compatibilidad en cualquier entorno y a los pocos recursos que consume en el sistema donde se instala.
Por lo tanto será necesario crear tres imágenes que darán lugar a los tres contenedores que albergarán la base de datos, el backend y el frontend respectivamente y que se unirán por medio de docker-compose. La Figura 5.3 representa el proceso llevado a cabo y que se explica a continuación.

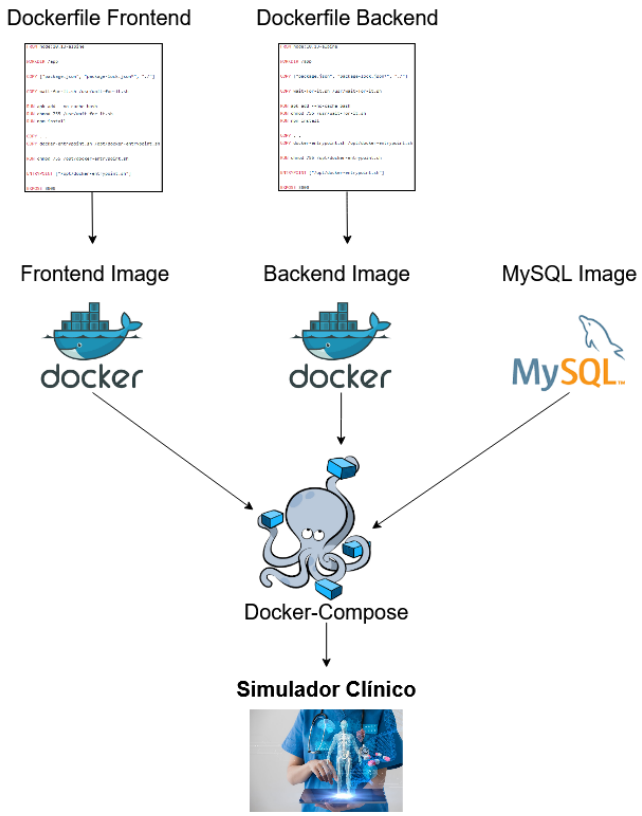

Figura 5.3: Despliegue del simulador clínico para traumatismos en Docker.

5.2.1. Imagen 1: Frontend

Se ha creado un contenedor Docker que albergará todo el código del frontend diseñado y comentado en el Capítulo 3 . Para ello se ha montado una imagen que se describe en el Dockerfile de la Figura 5.4. En este archivo de texto aparecen diferentes instrucciones Docker:

- **From:** indica la imagen base sobre la que se construirá el contenedor.
- **Workdir:** sirve para acceder al directorio de trabajo que venga escrito tras la instrucción. En el caso de que el directorio no exista será creado.
- **Copy:** su función es copiar un archivo desde la máquina física del host y añadirlos a un directorio del contenedor.
- **Run:** la instrucción Run da la posibilidad de ejecutar instrucciones en el contenedor, propias de la imagen base sobre la que se ha construido.
- **Entrypoint:** se utiliza para ejecutar las instrucciones que se llevarán a cabo tras el arranque del contenedor. Es decir cada vez que se ejecute un comando en el contenedor se estará llamando al Entrypoint.
- **Expose:** gracias a este comando es posible abrir e indicar los puertos TCP/IP a través por los cuales se pueden acceder a los servicios del contenedor creado.

```
FROM node:10.13-alpine

WORKDIR /app

COPY ["package.json", "package-lock.json*", "./"]

COPY wait-for-it.sh /usr/wait-for-it.sh

RUN apk add --no-cache bash
RUN chmod 755 /usr/wait-for-it.sh
RUN npm install

COPY . .
COPY docker-entrypoint.sh /opt/docker-entrypoint.sh

RUN chmod 755 /opt/docker-entrypoint.sh

ENTRYPOINT ["/opt/docker-entrypoint.sh"]

EXPOSE 3000
```

Figura 5.4: Frontend Dockerfile

Analizando la Figura 5.4, se puede ver que se ha elegido Node Alpine como imagen base. Alpine es una distribución Linux muy pequeña que ocupa aproximadamente 5 MB de capacidad [49]. En el caso del simulador clínico basta con tener instalado Node.js, es por ello que se utiliza la imagen Node Alpine que ya se encuentra disponible, evitando instalar numerosos paquetes que vienen por defecto en otras distribuciones Linux que no se utilizarán y ocuparán espacio.
Continuando con el análisis, se observa que se ha creado un directorio donde se copian las archivos JSON en el que están incluidas las dependencias y librerías necesarias, y que se instalarán con la ejecución de los comandos correspondientes. Por último se habilita el puerto 3000, por el que se podrá acceder a la aplicación.

5.2.2. Imagen 2: Backend

Al igual que en el apartado anterior, también se ha escrito un Dockerfile para el servidor del simulador diseñado con el fin de desplegarlo en un contenedor Docker. Como se puede apreciar en la Figura 5.5 que representa al Dockerfile del backend, la única diferencia con la Figura 5.4 es el número del puerto utilizado para acceder al servidor, siendo éste el 8080, matizando que se refiere al puerto del contenedor, no al de la máquina física en la que se ejecuta el contenedor. Una imagen está formada por distintas capas que se van instalando de manera independiente. Las capas son cada una de las instrucciones Docker que están descritas en el archivo Dockerfile. Si algunas de estas instrucciones se modifican, Docker tan solo reinstalará la capa que ha sido modificada, dejando las demás capas intactas. Esto hace que la instalación de una imagen sea eficiente, ocupe poco espacio y no lleve demasiado tiempo.

```
FROM node:10.13-alpine

WORKDIR /app

COPY ["package.json", "package-lock.json*", "./"]
COPY wait-for-it.sh /usr/wait-for-it.sh

RUN apk add --no-cache bash
RUN chmod 755 /usr/wait-for-it.sh
RUN npm install

COPY . .
COPY docker-entrypoint.sh /opt/docker-entrypoint.sh

RUN chmod 755 /opt/docker-entrypoint.sh

ENTRYPOINT ["/opt/docker-entrypoint.sh"]

EXPOSE 8080
```

Figura 5.5: Backend Dockerfile

También en ambas figuras aparece el archivo *wait-for-it.sh*. Este archivo es un script que se ejecutará en el entrypoint y que sirve para esperar a que un determinado contenedor esté arrancado por completo, y el uso que se le ha dado en el simulador se explicará en la sección 5.2.4.

5.2.3. Imagen 3: Base de datos

Para desplegar una base de datos MySQL en Docker se ha utilizado una imagen ya creada que se encuentra en el registro público Docker Hub. Es por ello que no ha sido necesario escribir un Dockerfile personalizando la imagen, y será reutilizada tal y como está desarrollada. Tan sólo habrá que añadir algunas variables de entorno que señalen el nombre de la base de datos a utilizar y la contraseña para acceder a ella.

5.2.4. Composición de las imágenes

Para unir las tres imágenes detalladas anteriormente se va a utilizar docker-compose, que ayudará a arrancar los tres contenedores correspondientes a cada una de las imágenes de manera conjunta. Para el correcto funcionamiento de la parte visual del simulador es necesario tener acceso a las diferentes APIs que manejan los datos de la simulación. Para que estas APIs puedan modificar y recoger datos es imprescindible que la base de datos esté conectada al backend. Por lo tanto el orden en el que se deberá arrancar la aplicación será primero el contenedor de la base de datos seguido del backend y, posteriormente, del frontend. Esto se consigue gracias al script *wait-for-it.sh*.

Para conseguir esté arranque orquestado mediante docker-compose es necerario escribir un archivo *docker-compose.yml* cuya estructura se muestra en la Figura 5.6. Este archivo está divido en servicios. Cada uno de los servicios representan a los tres contenedores Docker que se ejecutan para dar lugar al simulador clínico. A continuación se destacan algunas características de la configuración reflejadas en la Figura 5.6:

- Para tener acceso a los diferentes recursos del simulador es necesario conocer los **puertos TCP/IP** que estarán habilitados desde la máquina física. Es por ello que en el *docker-compose.yml* aparecen los mapeos de estos puertos, indicando en primer lugar el puerto accesible en el host seguido del puerto del contenedor.
- Para la configuración de la base de datos es necesario conocer la contraseña para acceder a ella y el nombre de la misma, para esto se utilizan las **variables de entorno**, escritas en el apartado *environment*.
- Por último, cabe destacar la forma que tiene el simulador de guardar los cambios que se producen en la base de datos, y por lo tanto de asegurar el contenido de esta. Esto se consigue gracias a los **volúmenes Docker** que se asignan a un cierto directorio del contenedor, en este caso en el directorio donde se almacena la información de la base de datos, y se guarda en la máquina física.

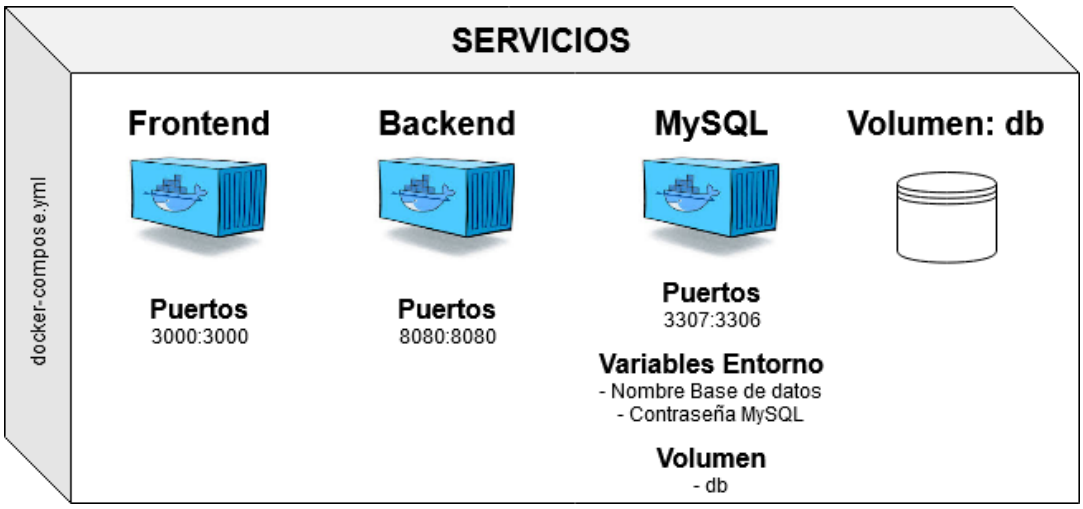

Figura 5.6: Estructura docker-compose.yml.

Así, tras el despliegue conjunto de los tres contenedores el simulador clínico estaría montado y listo para su uso.

Capítulo 6

Conclusiones y líneas futuras

6.1. Conclusiones

En este Trabajo Fin de Grado se ha desarrollado un simulador clínico web alcanzando los objetivos propuestos al inicio del presente documento y la consecución de una aplicación web didáctica para que los profesionales médicos del Hospital Universitario La Paz puedan entrenar los diferentes protocolos de actuación ante lesiones traumáticas, realizando pruebas diagnósticas que revelen la gravedad del trauma y llevando a cabo los diferentes tratamientos que se aplicarían para estabilizar al paciente.

Dos perfiles clínicos diferentes, un instructor y un estudiante, pueden interactuar con la aplicación a través de su interfaz. El instructor puede crear un caso clínico personalizado estableciendo las constantes vitales del paciente virtual, el tipo de traumatismo, el estado mental y el tiempo del que dispone el estudiante para realizar la simulación con éxito, teniendo en cuenta la gravedad del traumatismo.
El estudiante puede acceder al caso clínico planteado por el instructor eligiendo las pruebas diagnósticas y los tratamientos a realizar sobre el paciente para su estabilización y recuperación. Las distintas acciones que el estudiante puede realizar sobre el paciente han sido configuradas para que su aplicación afecte al estado de salud del paciente haciendo variar sus constantes.
Se ha conseguido que la herramienta desarrollada para el entrenamiento de los profesionales sea didáctica, corrigiendo y resaltando el trabajo realizado. Para ello se ha desarrollado un informe, que estará disponible al finalizar la simulación, en el que se recogen todas las acciones que el estudiante ha llevado a cabo desde el inicio de la simulación hasta el final, informando sobre el estado y la evolución del paciente. Este informe es descargado por el instructor para que pueda valorar el trabajo desarrollado por el estudiante en el transcurso de la simulación, siendo capaz de remitir una crítica que refleje los aciertos y los fallos del estudiante.

Finalmente, se ha realizado un despliegue de la aplicación en el que el simulador puede ser instalado en cualquier terminal, evitando problemas de compatibilidad con los programas que se utilizan para que la aplicación funcione correctamente.

6.2. Líneas futuras

Tras un primer contacto con el simulador por parte de los profesionales del Hospital Universitario La Paz, se han establecido una serie de mejoras y próximos desarrollos que ayudarán a conseguir un simulador clínico más completo y realista:

- Organizar las acciones que puede llevar a cabo el estudiante en distintas secciones de manera que las acciones queden estructuradas.
- Diseñar un maniquí físico sobre el que puedan realizarse algunas de las maniobras y tratamientos que están representados en el conjunto de las acciones de la aplicación web, de manera que el simulador adquiera un grado de realismo mayor. Además de obtener un simulador más completo en el que el estudiante pueda aprender tanto qué tipos de cuidados y pruebas se deben realizar en el tratamiento de un traumatismo grave, como la técnica de la aplicación de los diferentes cuidados de una manera práctica.
- Preparar un protocolo de comunicación para recibir las señales que provengan del maniquí físico para que las acciones llevadas a cabo manualmente por parte del estudiante, tengan consecuencia directa en la saludo del paciente virtual.
- Mejorar la experiencia de usuario creando un carrusel explicativo a modo de tutorial que trate de enseñar el funcionamiento de la aplicación.
- Añadir y modificar respuestas y acciones que ayuden a mejorar la evolución del paciente en tiempo real.

References

[1] OMS. Las 10 principales causas de defunción. https://www.who.int/es/news-room/fact-sheets/detail/the-top-10-causes-of-death, 2018. [Online]; accedido 15/05/2020.

[2] Eurostat. Estadísticas sobre causas de muerte. https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Causes_of_death_statistics/es, 2016. [Online]; accedido 15/05/2020.

[3] Instituto Nacional de Estadística. Defunciones según causas externas. https://www.ine.es/jaxiT3/Datos.htm?t=7947!tabs-tabla, 2018. [Online]; accedido 15/05/2020.

[4] Grupo de Trabajo de Trauma y Neurointensivismo SEMICYUC. Epidemiología del trauma grave en españa. registro de trauma en uci (retrauci). fase piloto. *Medicina Intensiva*, 40(6):327–347, 2016.

[5] Generación Elsevier. Escala de coma de glasgow: tipos de respuesta motora y su puntuación. https://www.elsevier.com/es-es/connect/medicina/escala-de-coma-de-glasgow, 2017. [Online]; accedido 20/05/2020.

[6] Miriam García. Mvc (modelo-vista-controlador): ¿qué es y para qué sirve? https://codingornot.com/mvc-modelo-vista-controlador-que-es-y-para-que-sirve, 2017. [Online]; accedido 24/05/2020.

[7] manufosela. Introducción práctica a docker y docker-compose. https://medium.com/@manufosela/introducci%C3%B3n-pr%C3%A1ctica-a-docker-y-docker-compose-f0dba8d7fd28, 2018. [Online]; accedido 29/05/2020.

[8] José Legido. Docker: Imágenes y vida de los contenedores. https://avanttic.com/blog/docker-imagenes-vida-contenedores/, 2017. [Online]; accedido 04/06/2020.

[9] L. Atutxa M. Zabarte F. Alberdi, I. García. Epidemiología del trauma grave. *Medicina Intensiva*, 38(9):580–588, 2014.

[10] Instituto Nacional de Estadística. Defunciones según la causa de muerte. https://www.ine.es/jaxiT3/Datos.htm?t=6609!tabs-tabla, 2018. [Online]; accedido 14/05/2020.

[11] SEMICYUC. La mortalidad por traumatismo grave ha aumentado hasta el 15 % en los últimos años. https://semicyuc.org/wp-content/uploads/2019/10/NP_La-mortalidad-por-traumatismo-grave-ha-aumentado-hasta-el-15.pdf.

[12] L.C. Hinojosa-Arco L. Ocaña-Wilhemi G. Moratalla-Cecilia, R.Gómez-Pérez. Protocolo del código trauma en hospital de nivel iii. *Cirugía Andaluza*, 30(1):107–112, 2019.

[13] Guevara O. Ruíz-Parra A, Ángel-Muller E. La simulación clínica y el aprendizaje virtual. tecnologías complementarias para la educación médica. *Clinical simulation and virtual learning. Complementary technologies for medical education*, 57(1):67–79, 2009.

[14] Marcia Corvetto. Simulación en educación médica: una sinopsis. *Revista médica de Chile*, 141(1):70–79, 2013.

[15] Andrea Davila-Cervantes. Simulacion en educacion medica. *Investigacion en Educacion Medica*, 3(10):100–105, 2014.

[16] Ana Beatriz Nates Reyes-Marta Deysi Pérez Arbolay Yanetsi Contreras Olive, Marllany Reyes Fournier. Los simuladores como medios de enseñanza en la docencia medica. *Revista Cubana de Medicina Militar*, 47(2), 2018.

[17] Ignacio del Moral José A. Riancho Javier Riancho, José M. Maestre. Simulación clínica de alto realismo: una experiencia en el pregrado. *Educación Médica*, 15(2):109–115, 2012.

[18] Noceda Bermejo JJ Pérez Lahiguera FJ. Atención al trauma grave. http://www.dep4.san.gva.es/contenidos/urg/reserv/archivos/protocolos/VIA 20POLITRAUMATISMO.pdf, 2014. [Online]; accedido 05/04/2020.

[19] Wikipedia. Traumatismo craneoencefálico. https://es.wikipedia.org/wiki/Traumatismo_craneoencef%C3%A1lico, 2020. [Online]; accedido 27/05/2020.

[20] Marta Papponetti. Escala de coma de glasgow. https://www.intramed.net/contenidover.asp?contenidoid=94304, 2019. [Online]; accedido 15/04/2020.

[21] Gamal Hamdan Suleiman M.D. Trauma craneoencefálico severo: Parte i. *Medicrit*, 2(7):107–148, 2005.

[22] Philbert Yuan Van. Introducción a los traumatismos abdominales. https://www.merckmanuals.com/es-us/hogar/traumatismos-y-envenenamientos/traumatismos-abdominales/introducci%C3%B3n-a-los-traumatismos-abdominales, 2017. [Online]; accedido 21/05/2020.

[23] D. Patricio Rodríguez Dr. P. David Lazo Dr. M. Felipe Undurraga Dr. Trauma de tórax. *Revista Médica Clínica Las Condes*, 22(5):617–622, 2011.

[24] María Teresa Arrieta González. Traumatismo pelvico: Hallazgo de imagen. *Revista Médica Sinergia*, 2(7):3–5, 2017.

[25] A. Bru Pomer L. Hernández FERRANDO. Fracturas pélvicas: una visión moderna. *Revista Española de Cirugía Osteoarticular*, 50(261):39–48, 2015.

[26] Paula Sotelo V Jose Ortega S Miguel Plaza de los Reyes Z Hernán Aguilera M Ricardo Sonneborn G Ricardo Espinoza G, Patricio Dietz W. Trauma arterial de extremidades: Resultados delmanejo por el cirujano no especialista. *Revista Chilena de Cirugía*, 54(3):225–230, 2002.

[27] MD Sharon Henry. *Apoyo Vital Avanzado en Trauma*, volume 4 of *10*. The name of the publisher, American College of Surgeons 633 N. Saint Clair Street Chicago, IL, 10 edition, 2018.

[28] José María Jover. Atls: 25 años de experiencia. *Cirugía Española*, 80(6):347–348, 2006.

[29] Toral Vázquez D Cuenca Solanas M Bermejo Aznárez S, Alted López E. Atención inicial al trauma grave. utilidad de la videograbación en la valoraciónde la calidad asistencial. *Trauma Fund MAPFRE*, 24(1):39–47, 2013.

[30] Luís Rafael Moscote Salazar. Trauma craneoencefálico atención inicial y manejo hospitalario. *Duazary: Revista internacional de Ciencias de la Salud*, 7(1):100–105, 2010.

[31] Karen Lissette Sánchez Anzules Gema Paola Zambrano Andrade Denisse Alejandra Pinela Baldeon, Tracy Tatiana Moran Lema. Abordaje en trauma cerrado de abdomen. *RECIMUNDO: Revista Científica de la Investigación y el Conocimiento*, 3(3):224–242, 2019.

[32] José Valverde Molina Ana M.ª González Fernández, Antonio Ramón Torres Torres. Traumatismo torácico, neumotórax, hemoptisis y tromboembolismo pulmonar. *NEUMOPED*, 1(1):189–209, 2017.

[33] A. García-Holgado F. J. García-Peñalvo, M. N. Moreno Garcia. Fundamentos de la vista de casos de uso. https://repositorio.grial.eu/bitstream/grial/1155/1/UML%20-%20Casos%20de%20uso.pdf, 2018. [Online]; accedido 28/05/2020.

[34] Yanette Díaz González Yenisleidy Fernández Romero. Patrón modelo-vista-controlador. *Revista Telem@tica*, 11(1):47–57, 2012.

[35] Damián Pérez Valdés. ¿qué es javascript? http://www.maestrosdelweb.com/que-es-javascript/, 2007. [Online]; accedido 5/06/2020.

[36] Javier Flores Herrera. Qué es html. https://codigofacilito.com/articulos/que-es-html, 2015. [Online]; accedido 5/06/2020.

[37] Ángel Robledano. Qué es css y para qué sirve. https://openwebinars.net/blog/que-es-css/, 2019. [Online]; accedido 5/06/2020.

[38] Miguel Angel Alvarez. Qué es react. por qué usar react. https://desarrolloweb.com/articulos/que-es-react-motivos-uso.html, 2019. [Online]; accedido 24/05/2020.

[39] Álvaro Fontela. ¿que es bootstrap? https://raiolanetworks.es/blog/que-es-bootstrap/, 2015. [Online]; accedido 4/06/2020.

[40] Judit Cabana. ¿cómo funciona node.js? https://www.drauta.com/que-es-nodejs-y-para-que-sirve, 2017. [Online]; accedido 25/05/2020.

[41] Uriel Hernández. Mvc (model, view, controller) explicado. https://codigofacilito.com/articulos/mvc-model-view-controller-explicado, 2015. [Online]; accedido 6/06/2020.

[42] Manuel Martínez Fernández. Crear una app nodejs con express generator. https://malnuer.es/js/crear-una-app-nodejs-con-express-generator/, 2015. [Online]; accedido 5/06/2020.

[43] Ángel Robledano. Qué es mysql: Características y ventajas. https://openwebinars.net/blog/que-es-mysql/, 2019. [Online]; accedido 03/06/2020.

[44] Gustavo B. ¿qué es mysql? explicación detallada para principiantes. https://www.hostinger.es/tutoriales/que-es-mysql/, 2019. [Online]; accedido 6/06/2020.

[45] Pastor Ramos. Qué es y para qué sirve sql. https://styde.net/que-es-y-para-que-sirve-sql/, 2018. [Online]; accedido 03/06/2020.

[46] Miguel Romanos. ¿qué es un orm? https://www.dreams.es/transformacion-digital/desarrolladores-paginas-web/que-es-un-orm, 2019. [Online]; accedido 03/06/2020.

[47] Arsys. ¿qué es docker y qué ventajas tiene trabajar con sus contenedores? https://www.arsys.es/blog/soluciones/docker-ventajas-contenedores/, 2019. [Online]; accedido 29/05/2020.

[48] Gigi Sayfan. Docker desde las bases: Comprendiendo imágenes. https://code.tutsplus.com/es/tutorials/docker-from-the-ground-up-understanding-images–cms-28165, 2017. [Online]; accedido 7/06/2020.

[49] Alpine Linux. alpine docker official images. https://hub.docker.com/$_{/}$*alpine*, 2020. [*Online*]; *accedido*12/05/2020.

Apéndice A

Aspectos éticos, económicos, sociales y ambientales

Tras finalizar el simulador clínico web interactivo para traumatismos que se ha desarrollado en este Trabajo Fin de Grado, se procede a analizar el impacto que puede tener este simulador o que haya podido tener su desarrollo con el fin de conocer las ventajas y desventajas que se pueden encontrar en los aspectos social, económico, ético y ambiental.

A.1. Aspecto social

El avance de la medicina supone una gran noticia para toda la población mundial. Este avance es una realidad, un estudio de la OMS en 2016 situaba la esperanza de vida de la población mundial en torno a los 72 años, y es que este dato ha seguido un crecimiento exponencial a lo largo de las últimas décadas. Particularmente, España se sitúa entre los tres primeros países con mayor esperanza de vida situándose ésta en torno a los 83 años de media.

Estos avances no son una mera casualidad, y en gran medida gracias a la tecnología. En estos momentos se utilizan numerosos recursos tecnológicos que ayudan a luchar contra problemas y enfermedades que se dan cada día. Gracias a este avance, hoy en día muchas enfermedades tienen un gran pronóstico. Las herramientas tecnológicas que han ayudado al avance abarcan numerosos ámbitos desde la detección de enfermedades hasta la implantación de una prótesis.

En este Trabajo Fin de Grado se ha desarrollado un simulador, y como no podía ser menos, estos mantienen una estrecha relación con la medicina. En el ámbito sanitario es imprescindible responder con rapidez ante cualquier situación. Por lo tanto, cualquier entrenamiento previo podría ayudar a tomar decisiones rápidas y eficaces. Este simulador se ha desarrollado con la idea y el deseo de que pueda formar parte de este avance y en consecuencia, ayudar también a todas las personas que en cualquier momento se puedan ver afectadas por una lesión traumática grave.

A.2. Aspecto económico:

En cuanto al impacto económico, un simulador médico puede ayudar a reducir ciertos gastos sanitarios. Gracias a los simuladores médicos, un profesional es capaz de entrenarse ante situaciones que puedan suceder en cualquier momento de su carrera profesional. Es decir, este entrenamiento le ayudará a cometer los mínimos fallos. Estos errores pueden suponer la necesidad de aplicar tratamientos médicos que ayuden a subsanarlos, aumentando el gasto sanitario, ya sea en material o en personal. Adicionalmente, un simulador puede reducir el gasto en personal ya que no sería necesaria la presencia de un profesional cualificado durante la simulación.

A.3. Aspecto ético

Este Trabajo Fin de Grado no tiene un impacto ético muy profundo. Cabría destacar la responsabilidad en el uso del propio simulador y por ello, la responsabilidad de tratar a un paciente virtual prácticamente de la misma manera que se haría con una persona real, actuando como si se tuviese la vida de una persona en juego y así lograr grandes resultados mejorando sus habilidades.

A.4. Aspecto ambiental

Actualmente una de las mayores preocupaciones que tiene la sociedad es el cuidado del medio ambiente. En un principio, el desarrollo de este simulador no tiene gran efecto en este ámbito. Pero se puede destacar que al tratarse de un simulador web, no se ha requerido la fabricación o el uso de algunos productos que puedan causar un impacto en el medio ambiente.

Apéndice B

Presupuesto económico

A lo largo de cinco meses se ha desarrollado este Trabajo Fin de Grado que finaliza con el diseño del simulador clínico gracias a la colaboración del Hospital Universitario La Paz y del departamento de Tecnología Fotónica y Bioingeniería. Este apéndice muestra la estimación del presupuesto económico que habría costado el desarrollo de este simulador, teniendo en cuenta tanto los recursos humanos como los recursos materiales.

- **Costes de personal**

  Para calcular los costes en el personal encargado del desarrollo del simulador se han estimado las horas de trabajo dedicadas a los distintos procesos que se han llevado a cabo, como se puede ver en la Tabla B.1.

	Horas
Estudio de lesiones traumáticas	30
Estudio de React	45
Estudio de Node.js	50
Estudio de Express	40
Estudio de Docker	45
Diseño del Frontend	75
Diseño del Backend	65
Despliegue en Docker	50
TOTAL	**400**

Tabla B.1: Horas estimadas del desarrollo.

Según el informe Presente y futuro del Ingeniero de Telecomunicación, elaborado conjuntamente por el Colegio Oficial de Ingenieros de Telecomunicación (COIT), la Asociación Española de Ingenieros de Telecomunicación y la consultora IDC el salario medio anual de un recién graduado y el de un profesional con aproximadamente 15 años es de 23,553 €y 51,220 €respectivamente. A partir de este salario y 2000 horas laborales estimadas anuales, se obtiene un sueldo de 11.80 €/hora y 25.60 €/hora respectivamente.

La Tabla B.2 muestra el gasto total por el personal que ha llevado a cabo el desarrollo del simulador clínico.

	Coste horario (€)	Horas	Total (€)
Ingeniero - Jefe de proyecto	25.60	30	768
Estudiante de ingeniería	11.80	400	4,720
TOTAL			**5,488**

Tabla B.2: Costes de personal.

- **Costes de recursos materiales**

Durante el diseño de este simulador clínico web se ha utilizado como herramienta principal un terminal Asus F541U con procesador i7, 8GB de RAM y 1TB de almacenamiento. Debido a que la duración del desarrollo de este Trabajo Fin de Grado es menor que el tiempo de vida de las herramientas utilizadas se desglosa un presupuesto teniendo en cuenta la amortización de estas herramientas y que aparece en la Tabla B.3.

	Tiempo de vida (años)	Uds.	Coste (€)	Amortización (€/mes)	Uso (meses)	Total (€)
Asus F541U	4	1	819.00	17.06	5	85.30
TOTAL						**85.30**

Tabla B.3: Costes de recursos materiales.

La Tabla B.4 muestra el desglose total de los gastos con IVA incluido.

	Coste
Costes de material	85.30 €
Costes de personal	5,488 €
Subtotal	5,573.30 €
IVA	1170.40 €
Total	**6,743.70 €**

Tabla B.4: Costes totales.

Apéndice C

Manual de usuario

La aplicación diseñada en este Trabajo Fin de Grado representa un simulador clínico web en el que interactúan dos tipos de usuarios. El primero, un instructor encargado de crear un caso clínico para que el segundo usuario, un estudiante, trate de resolverlo. A continuación se presentarán las situaciones que puede tener un usuario que quiere interactuar con esta aplicación, facilitando y guiando al usuario que quiera usar el simulador por primera vez.

C.1. Registro del usuario

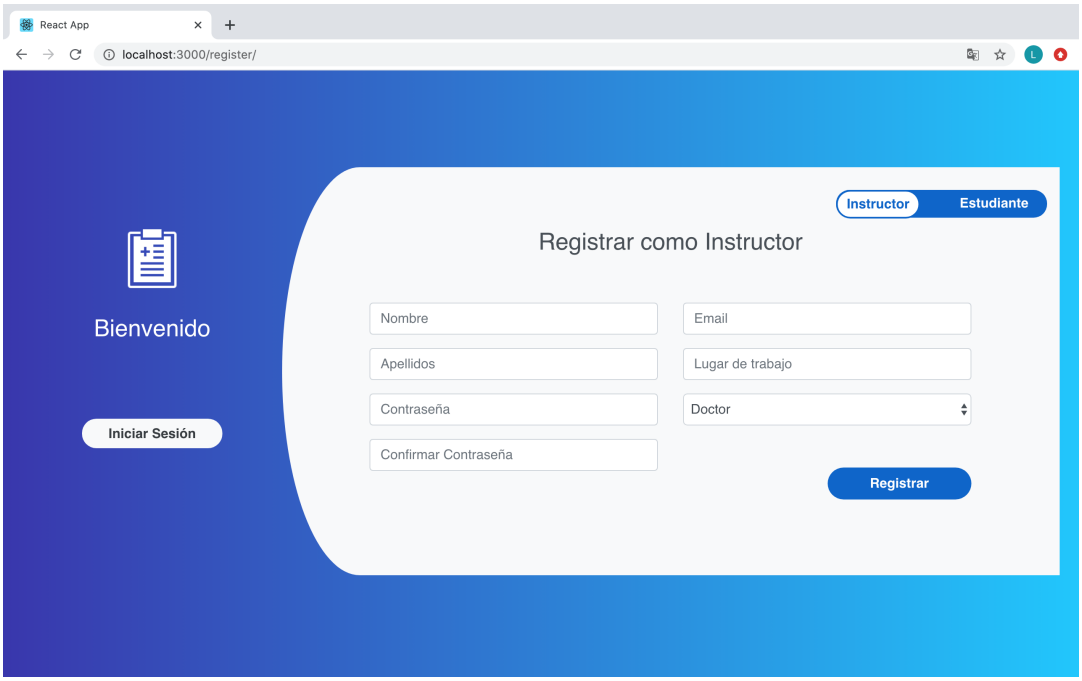

Figura C.1: Registro de un usuario en la aplicación.

En primer lugar, todo nuevo usuario que instale la aplicación se tendrá que registrar para poder tener acceso a las diferentes funcionalidades. Como se ha dicho hay dos tipos de usuarios, por lo tanto en el registro se da la opción de inscribirse como instructor o como estudiante. Además de conocer el tipo de usuario que se registrará habrá que conocer su email, nombre, apellidos, centro hospitalario donde trabaja o universidad donde estudia así como su ocupación (doctor, residente o estudiante). Además debe añadir una contraseña para poder acceder a la aplicación. Todo esto se muestra en la Figura C.1.

C.2. Instructor

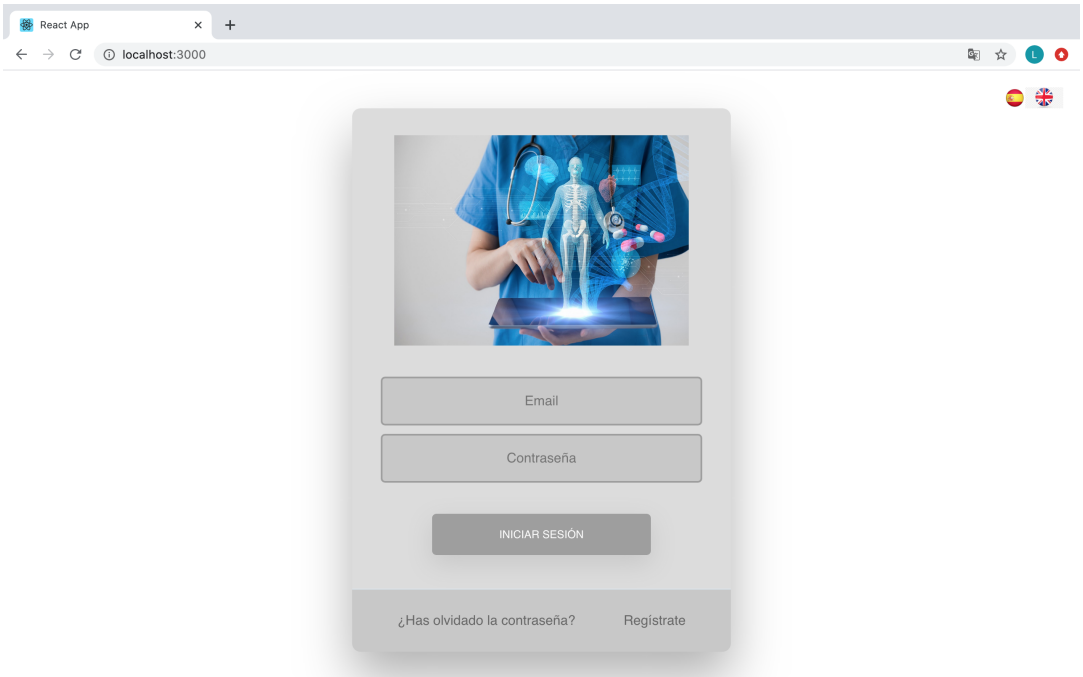

Figura C.2: Inicio de sesión de la aplicación.

De aquí en adelante se diferenciará entre instructor y estudiante, ya que no disponen de las mismas funcionalidades. Como se ve en la Figura C.2, esta es la vista en la que el usuario introducirá sus credenciales para poder acceder a la aplicación, entendiendo en esta sección que se trata de un instructor. Una vez se inicie sesión, aparece un menú principal en el que el instructor podrá elegir entre crear un nueva simulación o ver un listado de las simulaciones creadas, tal y como se puede ver en la Figura C.3.

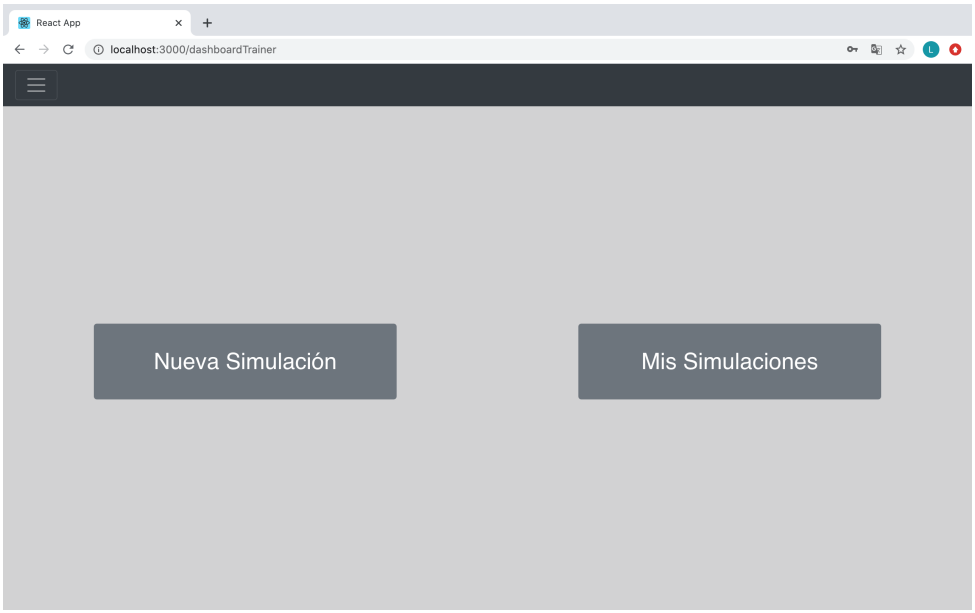

Figura C.3: Menú principal del instructor.

C.2.1. Nueva simulación

En esta pantalla el instructor diseñará un caso clínico personalizado que le será representado al estudiante para que lleve a cabo la simulación. Los casos clínicos se pueden dividir dependiendo de la zona donde se haya producido el traumatismo, que debe ser indicada en esta vista. Además de la zona de la lesión, el instructor deberá de introducir las constantes vitales y el estado del paciente en el momento de la simulación. Además se debe asignar la simulación a un estudiante y establecer un tiempo límite de duración de la simulación. Esta vista se representa en la Figura C.4.

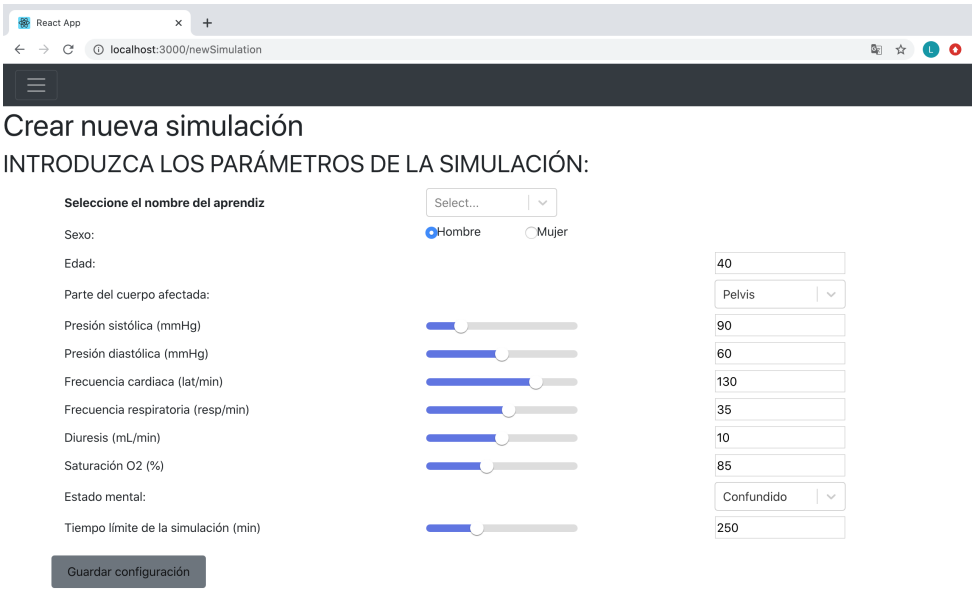

Figura C.4: Creación de un caso clínico.

C.2.2. Lista de simulaciones

El instructor también podrá acceder a la lista de simulaciones que él mismo haya creado para diferentes estudiantes, según muestra la Figura C.5. Desde aquí puede conocer si la simulación ya se ha realizado, puesto que una vez finalizada la simulación estará disponible un informe que recoge los pasos seguidos por el estudiante durante la simulación. Este informe podrá ser descargado por el instructor para poder evaluar la propia simulación y poder dar una realimentación al estudiante.

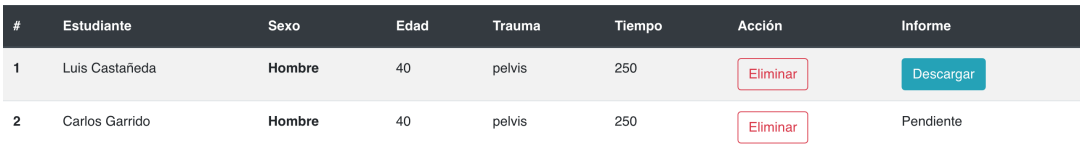

Figura C.5: Lista de simulaciones creadas por el instructor.

C.3. Estudiante

Situándose de nuevo en la vista de inicio de sesión de la Figura C.2, ahora se continúa con el inicio de un usuario que actúe como estudiante. Tras autenticarse, el estudiante accederá a un menú en el que podrá dirigirse a la lista de simulaciones creadas para ese estudiante.

C.3.1. Lista de simulaciones

La lista de la Figura C.6 muestra aquellas simulaciones que ha realizado o tiene que realizar el estudiante. Las que ya han sido realizadas tienen disponible el informe correspondiente al desarrollo de la propia simulación, y se podrá acceder a los comentarios que haya podido dejar el instructor que servirán de retroalimentación al estudiante. El estudiante también podrá acceder a aquellas simulaciones que tenga aún pendiente.

Figura C.6: Lista de simulaciones del estudiante.

C.3.2. Simulación

El estudiante accederá a la pantalla de la simulación que refleja la Figura C.7. En el momento que desee podrá pulsar para iniciar la simulación. En ese momento las gráficas mostrarán la representación de un monitor de constantes vitales del paciente virtual. En la parte superior izquierda aparece un maniquí que representa el paciente monitorizado y que irá cambiando su aspecto a medida que se le apliquen los tratamientos correspondientes. Estos tratamientos se podrán aplicar accionando cualquiera de los botones que aparecen debajo del maniquí y que representan las distintas pruebas y tratamientos que se pueden realizar al paciente. La aplicación de estas acciones sobre el paciente tendrán una consecuencia directa en el estado de éste y en el monitor de constantes se podrá ver la evolución. En la parte superior se sitúa el tiempo que queda disponible para terminar la simulación. Además se podrá adelantar este tiempo cuando no se pueda realizar ninguna acción más al paciente. La simulación finalizará cuando el tiempo se agote o el estudiante crea oportuno acabar la simulación considerando que el paciente se encuentra estabilizado.

Figura C.7: Simulación de un caso clínico.

C.3.3. Informe Final

Tras finalizar una simulación se generará automáticamente un informe, representado en la Figura C.8, en el que se recojan los pasos seguidos por el estudiante a lo largo de la simulación. El informe muestra los tratamientos y las pruebas que se han realizado sobre el paciente. Cada una de estás informaciones va acompañada del minuto en el que se han producido y de las constantes vitales que tenía el paciente en ese momento, conociendo la frecuencia cardiaca, la frecuencia respiratoria, la presión arterial, la saturación de oxígeno, la diuresis, la pérdida sanguínea y la temperatura corporal del paciente. Además al final del informe se adjuntan los resultados de las pruebas realizadas. Con todo ello, el instructor puede conocer cómo se ha desarrollado la simulación y cómo se ha desenvuelto el estudiante para poder dar su evaluación sobre lo ocurrido.

Informe Final

Estudiante: Luis Castañeda
Instructor: Blanca Larraga
Duración de la simulación: 212 minutos 41 segundos
Límite de tiempo: 250 minutos
Constantes iniciales del paciente: 130 puls/min, 35 resp/min, 90 mmHg, 60 mmHg, 85% SatO2 10 mL/min, 200 mL, 34.2 °C

Desarrollo de la simulación:

- Tiempo 0:20 **Colocar collarín cervical**
  130 puls/min, 35 resp/min, 90 mmHg, 60 mmHg, 85 % SatO2, 10 mL/min, 200 mL, 34.2 °C
- Tiempo 0:50 **Diálogo**
  130 puls/min, 35 resp/min, 90 mmHg, 60 mmHg, 85 % SatO2, 10 mL/min, 200 mL, 34.2 °C
- Tiempo 1:27 **Inspeccionar vía aérea**
  130 puls/min, 35 resp/min, 90 mmHg, 60 mmHg, 85 % SatO2, 10 mL/min, 200 mL, 34.2 °C
- Tiempo 2:44 **Limpiar vía aérea**
  130 puls/min, 35 resp/min, 90 mmHg, 60 mmHg, 85 % SatO2, 10 mL/min, 200 mL, 34.2 °C
- Tiempo 4:02 **Colocar mascarilla de oxígeno**
  129 puls/min, 33 resp/min, 90 mmHg, 60 mmHg, 86 % SatO2, 10 mL/min, 200 mL, 34.2 °C
- Tiempo 5:13 **Radiografía de tórax**
  128 puls/min, 30 resp/min, 91 mmHg, 60 mmHg, 88 % SatO2, 10 mL/min, 200 mL, 34.2 °C
- Tiempo 32:54 **Valoración CGS**
  121 puls/min, 26 resp/min, 94 mmHg, 61 mmHg, 92 % SatO2, 9 mL/min, 200 mL, 34.2 °C
- Tiempo 34:10 **Tomar temperatura**
  121 puls/min, 26 resp/min, 94 mmHg, 61 mmHg, 92 % SatO2, 9 mL/min, 200 mL, 34.2 °C
- Tiempo 35:48 **Colocar manta térmica**
  120 puls/min, 26 resp/min, 94 mmHg, 61 mmHg, 92 % SatO2, 9 mL/min, 200 mL, 34.2 °C

Figura C.8: Fragmento del informe generado tras la simulación.

Apéndice D

Manual del desarrollador

Como se ha detallado a lo largo de este documento, la aplicación diseñada se ha divido en tres partes, el backend, el frontend y el despliegue conjunto de estos en Docker. Para llevar a cabo estos desarrollos es imprescindible instalar los siguientes entornos que permiten el diseño de estas partes:

- **Node.js** versión 10.x o superior, que puede ser descargada a través de la pagina oficial `https://nodejs.org/es/`. Posteriormente se instalará el entorno Express de Node.js que albergará el servidor y la librería React encargada del desarrollo de la parte visual del simulador.
- El gestor de base de datos **MySQL** encargado de la creación de la base de datos en la que se albergarán los datos correspondientes a la simulación y a los usuarios que interactúan en la página oficial `https://dev.mysql.com/downloads/`.
- **Docker Engine** y **Docker Compose**, que se descargarán por medio de las páginas oficiales `https://docs.docker.com/engine/install/` y `https://docs.docker.com/compose/install/` respectivamente siguiendo las instrucciones que se detallan.
- **Visual Studio Code** que ha sido el entorno elegido para escribir y desarrollar todo el código de la aplicación que implementa el simulador clínico utilizado en la instalación de los servicios que se detallan en las siguientes secciones.

En realidad, la instalación de Node.js no es necesaria ya que gracias al despliegue de la aplicación en Docker, éste contará con todos los requisitos que requiera el despliegue y que se configuran durante el diseño de las imágenes Docker, y precisamente se contará con una versión operativa tanto de Node.js como de MySQL. Aún así, se ha querido destacar la instalación de ambos ya que durante el desarrollo del simulador son imprescindibles para que el simulador pueda funcionar sin utilizar Docker, arrancando manualmente la base de datos, el servidor y la aplicación en React, tal y como se explica en este apéndice.

El código fuente de la aplicación está disponible en `https://github.com/luiscastanedalopez/trauma-simulator.git`. Tras su descarga, sólo hay que abrir un terminal en el directorio recien descargado e introducir:

```
$ sudo docker-compose up --build
```

A continuación, la aplicación estará operativa en `http://localhost:3000`.

Ahora se detalla el desarrollo de la aplicación explicando el contenido del directorio del código fuente, en el que encontramos dos directorios nombrados como backend y frontend, que albergan el código del servidor y de la parte visual, y el archivo *docker-compose.yml* encargado del despliegue de la aplicación en Docker.

D.1. Backend

En el directorio backend se encuentra el directorio *src* que contiene todo el código referido al servidor y a la base de datos y cuenta con los siguientes directorios y ficheros:

- **App.js:** alberga las configuraciones que hacen posible el arranque del servidor en el entorno Express. Entre esas configuraciones está elegir el puerto TCP/IP en el que se encontrará accesible el servidor de la aplicación o la conexión con las rutas de las distintas APIs.

- **Controllers:** este directorio contiene cada uno de los archivos Javascript que forman las diferentes APIs de la aplicación.

- **Routes:** este directorio contiene los archivos Javascript que mapean cada una de las rutas establecidas a la API correspondiente desarrollada en el directorio *controllers*.

- **Model:** directorio que contiene los archivos Javascript con los detalles para la creación y configuración de la base de datos. Por un lado el archivo *database.js*, que facilita la conexión con la base de datos que se creará, escribiendo la información del gestor elegido, la contraseña y el nombre de la base de datos. Por otro lado, cada tabla que se quiere crear en la base de datos está representada en un archivo Javascript de este directorio, donde se incluirán los diferentes atributos y sus tipos que forman cada tabla.

Además del directorio *src*, se quieren resaltar otros directorios y archivos que también son importantes en el desarrollo:

- **package.json:** archivo JSON que contiene las características del servidor. En él se guardan el nombre de las dependencias que se han ido instalando a lo largo del desarrollo, así como las instrucciones que arrancar los scripts necesarios para la configuración del servidor.

- **node_modules:** directorio que contiene todas las dependencias instaladas.

- **migrations:** directorio que contiene la información de cada uno de los modelos de las tablas de la base de datos que se han definido, para su creación en la base de datos.

- **seeders:** directorio que contiene archivos Javascript en los que se escribe el contenido que se quiere introducir en las tablas de datos creadas que se deseen para rellenarlas de información.
- **Dockerfile:** archivo de configuración Docker en el que se adapta el entorno deseado para la instalación y arranque del servidor.

A continuación se detalla el proceso seguido para la creación de los archivos mencionados, dividiendo el proceso en la instalación del entorno Express para el arranque del servidor y en la configuración de la base de datos MySQL.

D.1.1. Express

Para llevar a cabo el diseño y la instalación del servidor se ha utilizado el entorno Express de Node.js siguiendo los siguientes pasos:

1. Crear un archivo package.json.

   ```
   $ npm init --yes
   ```

2. Instalar las dependencias express, mysql2, sequelize y sequelize-cli gracias a las cuales este servidor podrá funcionar.

   ```
   $ npm install nombreDependencia --save
   ```

   Al instalar la aplicación en un terminal diferente, en su primer arranque se deberán instalar todas las dependencias que se han utilizado en su desarrollo, por medio del comando:

   ```
   $ npm install
   ```

   Así todas las dependencias quedarán guardadas en el directorio *node_modules*.

3. Crear el directorio *src* y desarrollar los archivos explicados anteriormente.

4. Arrancar el servidor por medio de nodemon, utilizado para ejecutar el archivo *App.js*, que contiene las configuraciones para dicho arranque. Además, facilita el desarrollo haciendo que cualquier cambio que se produzca se actualice en tiempo real mientras el servidor está corriendo. Se instala de la siguiente manera:

   ```
   $ npm install --g nodemon
   ```

   Además se debe añadir un script en el package.json que sirva de arranque de la aplicación con nodemon.

   ```
   "dev": "nodemon src/App.js"
   ```

5. Por último, inicializar el servidor con el comando:

   ```
   $ npm run dev
   ```

D.1.2. Base de datos MySQL

Para instalar y crear la base de datos se ha utilizado el ORM Sequelize y para ello se han seguido los siguientes pasos en el directorio src/model/:

1. Escribir el archivo de configuración *mysql.js*, con la configuración de la base de datos.

2. Escribir los archivos Javascript correspondientes a cada una de las tablas de la base de datos.

3. Crear el fichero de migración correspondiente a cada uno de los modelos, por medio de la siguiente instrucción:

```
$ ./node_modules/.bin/sequelize migration:create
  --name CreateTableName
```

4. Crear el fichero Javascript de semillas para cada uno de los modelos, por medio de la siguiente instrucción:

```
$ ./node_modules/.bin/sequelize migration:create
  --name FillTableName
```

5. Rellenar adecuadamente los ficheros de migraciones con los datos de cada modelo y rellenar los ficheros de semillas que se deseen con la información que se quiera almacenar en la base de datos.

Una vez creados e instalados tanto el servidor como el modelo se procede al arranque del servidor y su conexión a la base de datos, para ello se deben de ejecutar los ficheros de migraciones y semillas que crean y rellenan la base de datos y por último arrancar el servidor.

```
$ npm install
$ npm run migrate
$ npm run seed
$ npm run dev
```

Estos comandos se configurarán en el Dockerfile para que una vez preparado Docker, pueda crear la base de datos con las tablas definidas y posteriormente arrancar el servidor.

D.2. Frontend

En el directorio *frontend* destacan los siguientes archivos y directorios gracias a los cuales se ha desarrollado la parte visual por medio de la librería React.

- **package.json:** archivo JSON que contiene las características de la aplicación React. En él se guardan el nombre de las dependencias que se han ido instalando a lo largo del desarrollo, así como las instrucciones que arrancan los scripts necesarios para iniciar la aplicación React.

- **yarn.lock:** archivo similar al anterior, en el que sólo se detallan las dependencias instaladas.
- **src:** directorio en el que se alojan todos los componentes React creados y que forman las diferentes vistas de la aplicación. Hay que destacar el fichero *App.js*, desde el cual se indica la ruta en la que se localizan cada una de las vistas diseñadas, y el fichero *i18.js*, que aloja la configuración para que el contenido de la aplicación pueda ser traducido al idioma deseado.
- **public:** directorio que alberga todos los archivos locales que ayudan a diseñar las vistas, principalmente imágenes y las traducciones a los idiomas deseados que se quieran mostrar en la aplicación.
- **node_modules:** directorio que contiene todas las dependencias instaladas, que facilitan y mejoran el diseño de los componentes.
- **Dockerfile:** archivo de configuración Docker en el que se configura el entorno deseado para la instalación y arranque de la aplicación React.

A continuación se muestran los pasos que se ha seguido para la elaboración de esta aplicación que dan lugar a los directorios y archivos detallados anteriormente:

1. Instalar de manera global el paquete *create-react-app* encargado de crear automáticamente el esqueleto de una aplicación react, destacando la creación de los directorios y archivos mencionados anteriormente *src, public* o *App.js* a partir de *yarn*, un gestor de paquetes parecido a *npm*, más eficiente y moderno.

   ```
   $ yarn global add create-react-app
   $ yarn create-react-app nombreApp
   ```

2. Instalar todas las librerías necesarias para el desarrollo de las vistas que muestran la aplicación. Aunque también se podrán instalar a medida que se vayan utilizando.

   ```
   $ yarn add nombreDependencia
   ```

   Al instalar la aplicación en un terminal diferente, en su primer arranque se deberán instalar todas las dependencias que se han utilizado en su desarrollo, por medio del comando:

   ```
   $ npm install
   ```

   Así todas las dependencias quedarán guardadas en el directorio *node_modules*.

3. Diseñar todos los componentes que darán lugar a cada una de las vistas del simulador.
4. Instalar la libreria Bootstrap y Reactstrap para mejorar el diseño de los componentes o incluso utilizar un componente ya creado.
5. Asignar en *App.js* a una ruta, la vista deseada formada por diferentes componentes, gracias a la dependencia *React Router*.

6. Instalar la librería axios, la cual se utilizará en los componentes que lo requieran para que el frontend se pueda comunicar con el backend, solicitando los recursos necesarios que se necesiten mostrar en las vistas o para almacenar en la base de datos, información recogida a través de la interacción con las diferentes vistas.

7. Escribir los archivos JSON que se incluirán en el directorio *public*, que contendrán las traducciones referidas al contenido de la aplicación.

Una vez finalizado el diseño de la aplicación en React y esté lista para ser usada tan solo habrá que introducir el siguiente comando y acceder a `http://localhost:3000` donde se ha configurado esta aplicación.

```
$ yarn start
```

Este comando también tendrá que ser configurado en el Dockerfile correspondiente al frontend, ya que una vez preparado el entorno en Docker, donde se ejecutará la aplicación, se deberá de ejecutar este comando.

D.3. Docker

Tras la instalación de Docker y el diseño del código tanto del frontend como del backend, se procede a la instalación de los contenedores Docker y su ejecución conjunta.

1. Crear los archivos Dockerfile del backend y del frontend mencionados anteriormente.

2. Preparar el entorno donde se ejecutarán el servidor y la aplicación React. Para ello se utilizará la imagen Docker *node:10.13-alpine*, que instalará Node.js escribiendo lo siguiente en el archivo Dockerfile:

```
FROM node:10.13-alpine
```

3. Se dará la siguiente instrucción para que el contenedor copie el código fuente tanto del directorio backend como del frontend, encontrandose disponible en el interior del contenedor tras su arranque.

```
WORKDIR /app
COPY ["package.json", "package-lock.json*", "./"]
```

4. Escribir en el Dockerfile las instrucciones que se quieran dar en la consola del contenedor para la instalación de las dependencias necesarias

```
RUN npm install
```

5. Escribir el script *docker-entrypoint.sh* para arrancar tanto el servidor como la aplicación React. En este script se introducen los comandos necesarios para el arranque que se han visto en las secciones anteriores en las que se detallaba la instalación de ambos servicios. Para acceder a este script se escribirá lo siguiente en el Dockerfile:

```
COPY docker-entrypoint.sh /opt/docker-entrypoint.sh
RUN chmod 755 /opt/docker-entrypoint.sh
ENTRYPOINT ["/opt/docker-entrypoint.sh"]
```

6. Será necesaria una imagen MySQL ya diseñada para la conexión del modelo creado con el gestor de bases de datos MySQL, esta imagen se establece en el archivo *docker-compose.yml*.

7. Escribir el fichero de configuración de *docker-compose.yml* para el arranque conjunto de los contenedores backend, frontend y MySQL, detallando los puertos TCP/IP accesibles para cada uno de los servicios.

8. Debido a que el arranque de los contenedores debe ser MySQL, backend y frontend por ese orden, se utiliza el script *wait-for-it.sh* durante el arranque de los contenedores backend y frontend para que ambos esperen a que el contenedor previo esté funcionando correctamente, antes de su inicio.

Cuando se tienen todos los archivos listos para el despliegue, se procede a dar la siguiente instrucción.

```
$ sudo docker-compose up - -build
```

Una vez se hayan arrancado todos los contenedores se podrá acceder al simulador diseñado dirigiéndose a `http://localhost:3000`.