Desarrollo de una aplicación web y base de datos para el seguimiento de pacientes con problemas de movilidad cervical

Borja González Díaz

Septiembre 2017

A toda mi familia y en especial a mi padre Peps.

**Resumen**

Con la llegada de las nuevas tecnologías al mercado la mayoría de ámbitos profesionales se están viendo forzados a adaptarse mediante la digitalización de todos sus sistemas, actualmente manuales y prácticamente obsoletos. Quedarse atascado en estos sistemas manuales significaría una gran desventaja a la hora de competir contra otros sistemas digitalizados, que son mucho más rápidos y eficientes, con una necesidad nula o mínima de mantenimiento. Esta disyuntiva también se aplica a los sistemas médicos, que se pueden beneficiar enormemente de la digitalización de la información relativa a los pacientes, permitiendo análisis más avanzados y una mayor agilidad para tratar a los pacientes.

Con un enfoque moderno a los métodos de visualización y recopilación de datos he construido una aplicación web con su correspondiente base de datos, que permiten al usuario acceder rápidamente a los datos de los pacientes permitiendo mostrar toda la información de manera sencilla y clara.

La aplicación web que he desarrollado se centra en el seguimiento de pacientes con problemas de movimiento cervical, específicamente en el rango de movimiento. El rango de movimiento cervical puede verse afectado o disminuido por varias causas como podrían ser los accidentes y más concretamente, el síndrome del latigazo vertical (SLC). Para poder medir el rango de movimiento del paciente, hay disponibles una serie de instrumentos que nos facilitan ésta tarea. Para este proyecto he utilizado un sensor inercial que proporciona una forma flexible y económica de medir la movilidad. Concretamente, el instrumento que vamos a utilizar se llama CROM (Cervical Range of Motioin Instrument).

Utilizando varios lenguajes de programación, plataformas y sistemas he conseguido satisfacer las necesidades dicha aplicación, cumpliendo así sus requisitos.

**Agradecimientos**

Tras estos años de carrera …

**Índice general**

Introducción 6

Funcionamiento de la aplicación web 6

Estado del arte 6

Diseño de web estático 6

HTML 6

CSS 7

NodeJS 7

Javascript 7

Socket.io 8

Papa Parse 8

Chart.js 9

Bases de Datos 9

SQLite 10

ACID 10

Diseño 11

Descripción del problema 11

Requisitos 12

Requisitos Funcionales 12

Requisitos no Funcionales 12

Casos de uso 13

Matriz de trazabilidad 17

Arquitectura del sistema 17

Aspecto de la aplicación web 17

Aspecto de la Base de datos 19

Estructura del archivo CSV 20

Especificación de la API 21

Implementación 21

# Introducción

## Funcionamiento de la aplicación web

El objetivo de esta aplicación web es facilitar el acceso y visualización de datos de movimientos cervicales de pacientes.

La página permite el acceso a una base de datos de pacientes donde se almacena un listado de pacientes y de movimientos. De dicha base de datos podemos consultar movimientos cervicales en todos los planos y observar la evolución con el tiempo de estos movimientos y comprobar si están dentro de unos parámetros de normalidad, dependiendo del sexo del paciente, ya que cada sexo tiene rangos de movilidad cervical distintos

En la sección de pacientes se listarán todos los pacientes disponibles, pudiendo visualizar su nombre y apellidos, a demás de tener la opción de añadir o borrar pacientes y podremos acceder a sus datos presionando en el icono del gráfico de cada paciente.

En la sección de datos de cada paciente, estará disponible un set de movimientos, que incluye los movimientos en los planos Transversal, Coronal y Sagital, con una fecha asociada que corresponde a la fecha de la medición. Se podrá consultar cada movimiento por separado en forma de gráfico. Además se podrá visualizar, en forma de gráfico, la evolución de cada movimiento a medida que pasa el tiempo. En el gráfico de evolución habrá disponible unos valores de normalidad para que sea posible observar si el paciente entra dentro de dichos valores. Se podrán añadir y borrar sets de movimientos, teniendo en cuenta que para añadir un set de movimientos habrá que seleccionar un archivo local que se del tipo CSV, ya que solo se aceptan este tipo de archivos, y asociar una fecha de medición a este set de movimientos con la hora incluida por si hubiese más de una medición diaria.

# 

# Estado del arte

## Diseño de web estático

### HTML

HTML, sigla en inglés de HyperText Markup Language (lenguaje de marcas de hipertexto), hace referencia al [lenguaje de marcado](https://es.wikipedia.org/wiki/Lenguaje_de_marcado) para la elaboración de [páginas web](https://es.wikipedia.org/wiki/P%C3%A1gina_web). Es un estándar que sirve de referencia del software que conecta con la elaboración de páginas web en sus diferentes versiones, define una estructura básica y un código (denominado código HTML) para la definición de contenido de una página web, como texto, imágenes, videos, juegos, entre otros. Es un estándar a cargo del [World Wide Web Consortium](https://es.wikipedia.org/wiki/World_Wide_Web_Consortium) ([W3C](https://es.wikipedia.org/wiki/W3C)) o Consorcio WWW, organización dedicada a la estandarización de casi todas las tecnologías ligadas a la web, sobre todo en lo referente a su escritura e interpretación. Se considera el lenguaje web más importante siendo su invención crucial en la aparición, desarrollo y expansión de la [World Wide Web](https://es.wikipedia.org/wiki/World_Wide_Web)(WWW). Es el estándar que se ha impuesto en la visualización de páginas web y es el que todos los navegadores actuales han adoptado.[1](https://es.wikipedia.org/wiki/HTML#cite_note-1)​

El lenguaje HTML basa su filosofía de desarrollo en la diferenciación. Para añadir un elemento externo a la página (imagen, vídeo, [script](https://es.wikipedia.org/wiki/Script), entre otros.), este no se incrusta directamente en el código de la página, sino que se hace una referencia a la ubicación de dicho elemento mediante texto. De este modo, la página web contiene solamente texto mientras que recae en el navegador web (interpretador del código) la tarea de unir todos los elementos y visualizar la página final. Al ser un estándar, HTML busca ser un lenguaje que permita que cualquier página web escrita en una determinada versión, pueda ser interpretada de la misma forma (estándar) por cualquier navegador web actualizado.

### CSS

**Hojas de Estilo en Cascada** (**C**ascading **S**tyle **S**heets) es el lenguaje utilizado para describir la presentación de documentos [HTML](https://developer.mozilla.org/es/docs/HTML) o [XML](https://developer.mozilla.org/es/docs/XML), esto incluye varios lenguajes basados en [XML](https://developer.mozilla.org/es/docs/XML) como son [XHTML](https://developer.mozilla.org/es/docs/XHTML) o [SVG](https://developer.mozilla.org/es/docs/SVG). CSS describe como debe ser renderizado el elemento estructurado en pantalla, en papel, hablado o en otros medios.

**CSS** es uno de los lenguajes base de la *Open Web* y posee una [especificación estandarizada](http://www.w3.org/Style/CSS/#specs) por parte del W3C. Desarrollado en niveles, CSS1 es ahora obsoleto, CSS2.1 es una recomendación y CSS3, ahora dividido en módulos más pequeños, está progresando en camino al estándar.

## NodeJS

**Node.js** es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor (pero no limitándose a ello) basado en el lenguaje de programación [ECMAScript](https://es.wikipedia.org/wiki/ECMAScript), asíncrono, con [I/O](https://es.wikipedia.org/wiki/I/O) de datos en una [arquitectura orientada a eventos](https://es.wikipedia.org/wiki/Programaci%C3%B3n_dirigida_por_eventos) y basado en el motor [V8](https://es.wikipedia.org/wiki/V8_(motor_JavaScript)) de Google. Fue creado con el enfoque de ser útil en la creación de programas de red altamente escalables, como por ejemplo, [servidores web](https://es.wikipedia.org/wiki/Servidor_web).

### Javascript

JavaScript es un [lenguaje de programación](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n) [interpretado](https://es.wikipedia.org/wiki/Int%C3%A9rprete_(inform%C3%A1tica)), dialecto del estándar [ECMAScript](https://es.wikipedia.org/wiki/ECMAScript). Se define como [orientado a objetos](https://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos), [basado en prototipos](https://es.wikipedia.org/wiki/Programaci%C3%B3n_basada_en_prototipos), [imperativo](https://es.wikipedia.org/wiki/Programaci%C3%B3n_imperativa), débilmente tipado y dinámico.

Se utiliza principalmente en su forma del [lado del cliente (*client-side*)](https://es.wikipedia.org/wiki/Lado_del_cliente), implementado como parte de un [navegador web](https://es.wikipedia.org/wiki/Navegador_web) permitiendo mejoras en la [interfaz de usuario](https://es.wikipedia.org/wiki/Interfaz_de_usuario) y [páginas web](https://es.wikipedia.org/wiki/P%C3%A1gina_web) dinámicas​ aunque existe una forma de JavaScript del [lado del servidor](https://es.wikipedia.org/wiki/Script_del_lado_del_servidor)(Server-side JavaScript o SSJS). Su uso en [aplicaciones](https://es.wikipedia.org/wiki/Aplicaci%C3%B3n_inform%C3%A1tica) externas a la [web](https://es.wikipedia.org/wiki/World_Wide_Web), por ejemplo en documentos [PDF](https://es.wikipedia.org/wiki/PDF), aplicaciones de escritorio (mayoritariamente [widgets](https://es.wikipedia.org/wiki/Widget)) es también significativo.

### Socket.io

Socket.io es una librería de JavaScript utilizada para aplicaciones web en tiempo real. Permite una comunicación bidireccional y en tiempo real entre cliente y servidor. Sockets.io está dividido en dos partes, una librería del cliente que se ejecuta en el navegador y una librería del servidor para Node.js. Ambas partes tienen prácticamente la misma API. Al igual que Node.js, Socket.io presenta una arquitectura orientada a eventos.

Generalmente Socket.io utiliza el protocolo WebSocket, y como opción alternativa utiliza el sondeo, pero siempre utiliza la misma interfaz.

Además de ofrecer la funcionalidad de WebSocket, Socket.io es capaz de emitir mensajes a varios sockets, almacenar datos asociados a cada cliente y E/S asíncrona.

### Papa Parse

Papa parse es un analizador sintáctico que convierte archivos de texto delimitados, principalmente archivos CSV, en estructuras de datos y viceversa. Papa Parse es el analizador más rápido en navegadores para JavaScript. Según la RFC 4180, Papa parse es el analizador sintáctico más fiable y por lo tanto el más recomendable. Posee las siguientes características:

* Muy fácil de usar.
* Parsea ficheros CSV en un entorno local o a través de la red directamente.
* Transmite archivos de gran tamaño (incluso a través de HTTP).
* El análisis sintáctico con la conversión se realizan, además, de forma inversa(JSON a CSV).
* Detección automática de delimitadores.
* Pausa, reanudación y anulación del parseo.
* Papa parse no tiene dependencias.
* Uno de los únicos analizadores que maneja correctamente los saltos de línea y las comillas.
* Puede convertir números y booleanos a sus tipos.
* Soporte del encabezado de fila.

### Chart.js

Chart.js es una plataforma de JavaScript que nos permite crear gráficos simples pero a la vez flexibles.

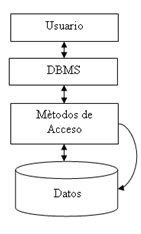
* A través del elemento <canvas> se consiguen crear gráficos simples.
* Con Chart.js podemos crear hasta 8 tipos de gráficos personalizables y con los que se puede interactuar.
* Posee un gran rendimiento en todos los navegadores actuales (IE9+).
* Redibuja los gráficos a la hora de ampliar o reducir sobre ellos para conseguir una escala perfecta de granularidad.
* Chart.js funciona muy bien a la hora de visualizar los gráficos en navegadores de otros dispositivos como tablets y móviles

## Bases de Datos

Una **base de datos** o **banco de datos** es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. En este sentido; una biblioteca puede considerarse una base de datos compuesta en su mayoría por documentos y textos impresos en papel e indexados para su consulta. Actualmente, y debido al desarrollo tecnológico de campos como la [informática](https://es.wikipedia.org/wiki/Inform%C3%A1tica) y la [electrónica](https://es.wikipedia.org/wiki/Electr%C3%B3nica), la mayoría de las bases de datos están en formato digital, siendo este un componente electrónico, por tanto se ha desarrollado y se ofrece un amplio rango de soluciones al problema del almacenamiento de datos.

Existen [programas](https://es.wikipedia.org/wiki/Programa_inform%C3%A1tico) denominados [sistemas gestores de bases de datos](https://es.wikipedia.org/wiki/Sistema_de_gesti%C3%B3n_de_bases_de_datos), abreviado SGBD (del inglés *database management system* o DBMS), que permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada. Las propiedades de estos DBMS, así como su utilización y administración, se estudian dentro del ámbito de la informática.

Las aplicaciones más usuales son para la gestión de empresas e instituciones públicas; También son ampliamente utilizadas en entornos científicos con el objeto de almacenar la información experimental.



### SQLite

SQLite es un sistema de gestión de [bases de datos relacional](https://es.wikipedia.org/wiki/Base_de_datos_relacional) (DBMS). Éste sistema es compatible con [ACID](https://es.wikipedia.org/wiki/ACID) y está contenido en una bibloteca en C que se podría considerar pequeña para ciertos casos (275kiB). El problema del tamaño se ha resuelto en la reciente versión · del sistemas que permite bases de datos de hasta 2 Terabytes. SQLite es un proyecto de [dominio público](https://es.wikipedia.org/wiki/Dominio_p%C3%BAblico" \o "Dominio público) creado por [D. Richard Hipp](https://es.wikipedia.org/w/index.php?title=D._Richard_Hipp&action=edit&redlink=1).

A diferencia de los [sistema de gestión de bases de datos](https://es.wikipedia.org/wiki/Sistema_de_gesti%C3%B3n_de_bases_de_datos" \o "Sistema de gestión de bases de datos) cliente-servidor, el motor de SQLite no es un proceso independiente con el que el programa principal se comunica. En lugar de eso, la biblioteca SQLite se enlaza con el programa pasando a ser parte integral del mismo. El programa utiliza la funcionalidad de SQLite a través de llamadas simples a subrutinas y funciones. Esto reduce la latencia en el acceso a la base de datos, debido a que las llamadas a funciones son más eficientes que la comunicación entre procesos. El conjunto de la base de datos (definiciones, tablas, índices, y los propios datos), son guardados como un sólo fichero estándar en la máquina host. Este diseño simple se logra bloqueando todo el fichero de base de datos al principio de cada transacción.

En su versión 3, como hemos mencionado antes, SQLite permite bases de datos de hasta 2 Terabytes de tamaño, y también permite la inclusión de campos tipo BLOB

### ACID

En bases de datos se denomina **ACID** a las características de los parámetros que permiten clasificar las [transacciones](https://es.wikipedia.org/wiki/Transacci%C3%B3n_de_base_de_datos) de los [sistemas de gestión de bases de datos](https://es.wikipedia.org/wiki/SGBD). Cuando se dice que es *ACID compliant* se indica -en diversos grados- que éste permite realizar transacciones.

En concreto **ACID** es un acrónimo de **A**tomicity, **C**onsistency, **I**solation and **D**urability: Atomicidad, Consistencia, Aislamiento y Durabilidad en español.

* [**Atomicidad**](https://es.wikipedia.org/wiki/Atomicidad)**:** Si cuando una operación consiste en una serie de pasos, bien todos ellos se ejecutan o bien ninguno, es decir, las transacciones son completas.
* [**Consistencia**](https://es.wikipedia.org/wiki/Consistencia_de_datos)**:** (*Integridad*). Es la propiedad que asegura que sólo se empieza aquello que se puede acabar. Por lo tanto se ejecutan aquellas operaciones que no van a romper las reglas y directrices de *Integridad* de la base de datos. La propiedad de consistencia sostiene que cualquier transacción llevará a la base de datos desde un estado válido a otro también válido. "La Integridad de la Base de Datos nos permite asegurar que los datos son exactos y consistentes, es decir que estén siempre intactos, sean siempre los esperados y que de ninguna manera cambien ni se deformen. De esta manera podemos garantizar que la información que se presenta al usuario será siempre la misma."
* [**Aislamiento**](https://es.wikipedia.org/wiki/Aislamiento_(ACID))**:** Esta propiedad asegura que una operación no puede afectar a otras. Esto asegura que la realización de dos transacciones sobre la misma información sean independientes y no generen ningún tipo de error.  Esta propiedad define cómo y cuándo los cambios producidos por una operación se hacen visibles para las demás operaciones concurrentes. El aislamiento puede alcanzarse en distintos niveles, siendo el parámetro esencial a la hora de seleccionar SGBDs.
* [**Durabilidad**](https://es.wikipedia.org/wiki/Datos_persistentes)**:** (*Persistencia*). Esta propiedad asegura que una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema y que de esta forma los datos sobrevivan de alguna manera.

Cumpliendo estos 4 requisitos un sistema gestor de bases de datos puede ser considerado *ACID Compliant*.

# Diseño

En este capítulo se intenta analizar en profundidad el problema que se pretende resolver, intentando abarcar la mayor cantidad de información posible y estudiando los principales aspectos que habrían de cubrirse en el desarrollo del proyecto. En este punto deberían surgir las características principales del sistema a desarrollar, por lo que se establecen también los casos de uso que deberá poder efectuar el sistema resultado del proyecto para cumplir los requisitos que se establezcan.

## Descripción del problema

El objetivo principal es construir una aplicación web con una base de datos asociada para el seguimiento de pacientes con problemas de movilidad cervical. Esta aplicación web permitirá el acceso de uno o más usuarios o clientes, y que estos puedan trabajar con la base de datos obteniendo información de los pacientes. Para describir el problema en cuestión voy a seguir un proceso que se divide en varios apartados. Empezaré con el diseño de la arquitectura del sistema para satisfacer las necesidades descritas previamente, seguiré con la implementación de dicho sistema, y terminaré describiendo el despliegue de las aplicaciones y pruebas.

## Requisitos

La fase de análisis incluye el estudio y la definición de los requisitos del sistema. Los requisitos definen en detalle los servicios que el sistema debería poder ofrecer, identificando las restricciones y características de este.

### Requisitos Funcionales

RF1 – Una vez dentro la aplicación web, el sistema debe mostrar una sección de inicio con una breve descripción de las distintas funcionalidades de la pagina.

RF2 – El sistema ofrecerá un listado de los clientes con su nombre, apellidos y un link para acceder a sus datos de movimiento.

RF3 – El usuario tendrá la opción de añadir pacientes, saltando un error en el caso de que el usuario no rellene algún dato solicitado.

RF4 – El usuario tendrá la opción de borrar pacientes, borrando así todos sus datos asociados.

RF5 – Para cada paciente, el sistema mostrará una tabla con los tres movimientos disponibles

RF6- El sistema ofrece la opción de mostrar la evolución de cada movimiento.

RF7 – El sistema debe proporcionar a los clientes la opción de añadir datos de movimientos con una fecha asociada.

RF8 - El sistema debe proporcionar a los clientes la opción de borrar datos de movimientos.

RF9 – El sistema deberá ser capaz de mostrar los datos de movimientos y los datos de evolución mediante gráficos animados e interactivos.

RF10 – El sistema permitirá observar la progresión del movimiento con el tiempo.

### Requisitos no Funcionales

El sistema debe permitir el acceso a la aplicación web desde cualquier dispositivo.

Solo se admitirán ficheros CSV.

Dentro de los gráficos de evolución el sistema marcará unos límites de normalidad dependiendo del sexo del paciente.

El sistema deberá ser capaz de almacenar los datos de los pacientes entre sesiones.

El sistema deberá reflejar los cambios realizados por un cliente conectado al servidor en el resto de clientes conectados.

## Casos de uso

Los casos de uso representan las diferentes posibles interacciones que pueden existir entre los diferentes actores dentro de un sistema. Los actores de un sistema son entidades externas al sistema que pueden interactuar con el. Los actores pueden ser humanos u objetos (servidores, aplicaciones, servicios...). Los actores que interactúan con nuestra aplicación son los siguientes:

* Usuarios del sistema a través del navegador de cualquier dispositivo.

CU1: Acceso a la página web.

El usuario introduce la dirección del servidor en su navegador accediendo instantáneamente al servidor.

Actores implicados: Usuario/cliente.

Precondiciones: El sistema debe estar correctamente funcionando.

Secuencia de acciones:

1. El cliente accede a la aplicación web introduciendo la dirección IP del servidor en su navegador.

CU2: Acceso a la pestaña de pacientes/Listado de pacientes.

El usuario accede a un listado de pacientes.

Actores implicados: Usuario/cliente.

Precondiciones: El sistema debe estar correctamente funcionando.

Secuencia de acciones:

1. El cliente accede a la aplicación web introduciendo la dirección IP del servidor en su navegador.
2. El cliente accede a la sección de pacientes mediante la pestaña de pacientes.
3. El navegador solicita la lista de pacientes a la base de datos.
4. El cliente recibe la lista de pacientes y se la presenta al usuario en una tabla.

CU3: Añadir un paciente.

Actores implicados: Usuario/cliente.

Precondiciones: El sistema debe estar correctamente funcionando.

Secuencia de acciones:

1. El cliente accede a la aplicación web introduciendo la dirección IP del servidor en su navegador.
2. El cliente accede a la sección de pacientes mediante la pestaña de pacientes.
3. El usuario introduce el nombre, apellidos y sexo del paciente y presiona el botón “Añadir”.
4. El navegador manda la petición, para añadir el paciente, al servidor.
5. El servidor añade el nuevo paciente a la base de datos, persistiendo los cambios.

Post-condiciones: El nuevo paciente tiene que ser mostrado inmediatamente en el listado de pacientes, sin que el usuario tenga que refrescar la página.

CU4: Borrar un paciente.

Actores implicados: Usuario/cliente.

Precondiciones: El sistema debe estar correctamente funcionando.

Secuencia de acciones:

1. El cliente accede a la aplicación web introduciendo la dirección IP del servidor en su navegador.
2. El cliente accede a la sección de pacientes mediante la pestaña de pacientes.
3. El navegador solicita la lista de pacientes a la base de datos.
4. El cliente recibe la lista de pacientes y se la presenta al usuario en una tabla.
5. El usuario elige el paciente a borrar y presiona el botón de borrado correspondiente, ya que cada paciente tiene su botón de borrado.
6. El navegador pregunta al usuario si desea realizar esta operación, teniendo en cuenta que borraría todos los datos de movimientos asociados a ese paciente.
7. El usuario acepta la operación, teniendo la opción de cancelarla.
8. El navegador manda la petición para borrar el paciente al servidor.
9. El servidor borra al paciente y todos sus datos de movimientos asociados de la base de datos, persistiendo los cambios.

Post-condiciones: La lista de pacientes ha de actualizarse, mostrando todos los pacientes menos el borrado. Se actualiza de forma automática. Todos los movimientos asociados a este paciente tienen que desaparecer.

CU5: Listado de movimientos de un paciente.

El usuario accede a un listado de movimientos del paciente.

Actores implicados: Usuario/cliente.

Precondiciones: El sistema debe estar correctamente funcionando.

Secuencia de acciones:

1. El cliente accede a la aplicación web introduciendo la dirección IP del servidor en su navegador.
2. El cliente accede a la sección de pacientes mediante la pestaña de pacientes.
3. El navegador solicita la lista de pacientes a la base de datos.
4. El cliente recibe la lista de pacientes y se la presenta al usuario en una tabla.
5. El cliente elige el paciente del que quiere mostrar sus datos de movimiento presionando el botón adecuado, ya que cada paciente tiene su propio botón para mostrar sus datos.
6. El navegador se posiciona en la pestaña de Datos.
7. El navegador solicita la lista de movimientos del paciente a la base de datos.
8. El navegador recibe la lista de movimientos y se la presenta al usuario en una tabla.

CU6: Añadir datos de movimiento.

El usuario añade un set de movimientos al paciente.

Actores implicados: Usuario/cliente.

Precondiciones: El sistema debe estar correctamente funcionando.

Secuencia de acciones:

1. El cliente accede a la aplicación web introduciendo la dirección IP del servidor en su navegador.
2. El cliente accede a la sección de pacientes mediante la pestaña de pacientes.
3. El navegador solicita la lista de pacientes a la base de datos.
4. El cliente recibe la lista de pacientes y se la presenta al usuario en una tabla.
5. El cliente elige el paciente del que quiere mostrar sus datos de movimiento presionando el botón adecuado, ya que cada paciente tiene su propio botón para mostrar sus datos.
6. El navegador se posiciona en la pestaña de Datos.
7. El usuario selecciona un archivo, que solo podrá ser del tipo CSV, y una fecha de medición del set movimientos y presiona el botón de “Añadir datos”.
8. El navegador manda la petición para añadir los datos de movimiento al servidor.
9. El servidor añade el set de datos de movimientos al la base de datos persistiendo los cambios.

Post-condiciones: El nuevo set de datos tiene que ser mostrado inmediatamente en el listado de datos de movimientos, sin que el usuario tenga que refrescar la página.

CU6: Borrar datos de movimiento.

El usuario borra un set de movimientos del paciente.

Actores implicados: Usuario/cliente.

Precondiciones: El sistema debe estar correctamente funcionando.

Secuencia de acciones:

1. El cliente accede a la aplicación web introduciendo la dirección IP del servidor en su navegador.
2. El cliente accede a la sección de pacientes mediante la pestaña de pacientes.
3. El navegador solicita la lista de pacientes a la base de datos.
4. El cliente recibe la lista de pacientes y se la presenta al usuario en una tabla.
5. El cliente elige el paciente del que quiere mostrar sus datos de movimiento presionando el botón adecuado, ya que cada paciente tiene su propio botón para mostrar sus datos.
6. El navegador se posiciona en la pestaña de Datos.
7. El navegador solicita la lista de movimientos del paciente a la base de datos.
8. El navegador recibe la lista de movimientos y se la presenta al usuario en una tabla.
9. El usuario elige los movimientos a borrar y presiona el botón de borrado correspondiente, ya que cada set de movimientos tiene su botón de borrado.
10. El navegador pregunta si desea realizar esta operación.
11. El usuario acepta la operación, teniendo la opción de cancelarla.
12. El navegador manda la petición para borrar los datos de movimiento al servidor.
13. El servidor borra el set de datos de movimiento del paciente de la base de datos y persiste los cambios.

Post-condiciones: La nueva lista de movimientos, sin el set de movimientos borrado, tiene que ser mostrada inmediatamente en el listado de movimientos, sin que el usuario tenga que refrescar la página.

CU7: Mostar datos del paciente.

El usuario muestra un movimientos del paciente.

Actores implicados: Usuario/cliente.

Precondiciones: El sistema debe estar correctamente funcionando.

Secuencia de acciones:

1. El cliente accede a la aplicación web introduciendo la dirección IP del servidor en su navegador.
2. El cliente accede a la sección de pacientes mediante la pestaña de pacientes.
3. El navegador solicita la lista de pacientes a la base de datos.
4. El cliente recibe la lista de pacientes y se la presenta al usuario en una tabla.
5. El cliente elige el paciente del que quiere mostrar sus datos de movimiento presionando el botón adecuado, ya que cada paciente tiene su propio botón para mostrar sus datos.
6. El navegador se posiciona en la pestaña de Datos.
7. El navegador solicita la lista de movimientos del paciente a la base de datos.
8. El navegador recibe la lista de movimientos y se la presenta al usuario en una tabla.
9. El usuario elige el movimiento (Sagital, Coronal, Transversal) con su fecha asociada presionando en el botón adecuado.
10. El navegador muestra por pantalla un gráfico con los datos requeridos.

Post-condiciones: El navegador tiene que borrar el contenido del gráfico anterior (en el caso de que exista) y graficar los datos requeridos.

CU8: Mostar datos de evolución del paciente.

El usuario muestra la evolución de un movimiento del paciente.

Actores implicados: Usuario/cliente.

Precondiciones: El sistema debe estar correctamente funcionando.

Secuencia de acciones:

1. El cliente accede a la aplicación web introduciendo la dirección IP del servidor en su navegador.
2. El cliente accede a la sección de pacientes mediante la pestaña de pacientes.
3. El navegador solicita la lista de pacientes a la base de datos.
4. El cliente recibe la lista de pacientes y se la presenta al usuario en una tabla.
5. El cliente elige el paciente del que quiere mostrar sus datos de movimiento presionando el botón adecuado, ya que cada paciente tiene su propio botón para mostrar sus datos.
6. El navegador se posiciona en la pestaña de Datos.
7. El navegador solicita la lista de movimientos del paciente a la base de datos.
8. El navegador recibe la lista de movimientos y se la presenta al usuario en una tabla.
9. El usuario elige la evolución del movimiento (Sagital, Coronal, Transversal) que quiere mostrar presionando el botón “Evolución del movimiento”.
10. El navegador muestra por pantalla un gráfico con los datos requeridos.

Post-condiciones: El navegador tiene que borrar el contenido del gráfico anterior (en el caso de que exista) y graficar los datos requeridos.

## Matriz de trazabilidad

## Arquitectura del sistema

### Aspecto de la aplicación web

#### Sección de Inicio



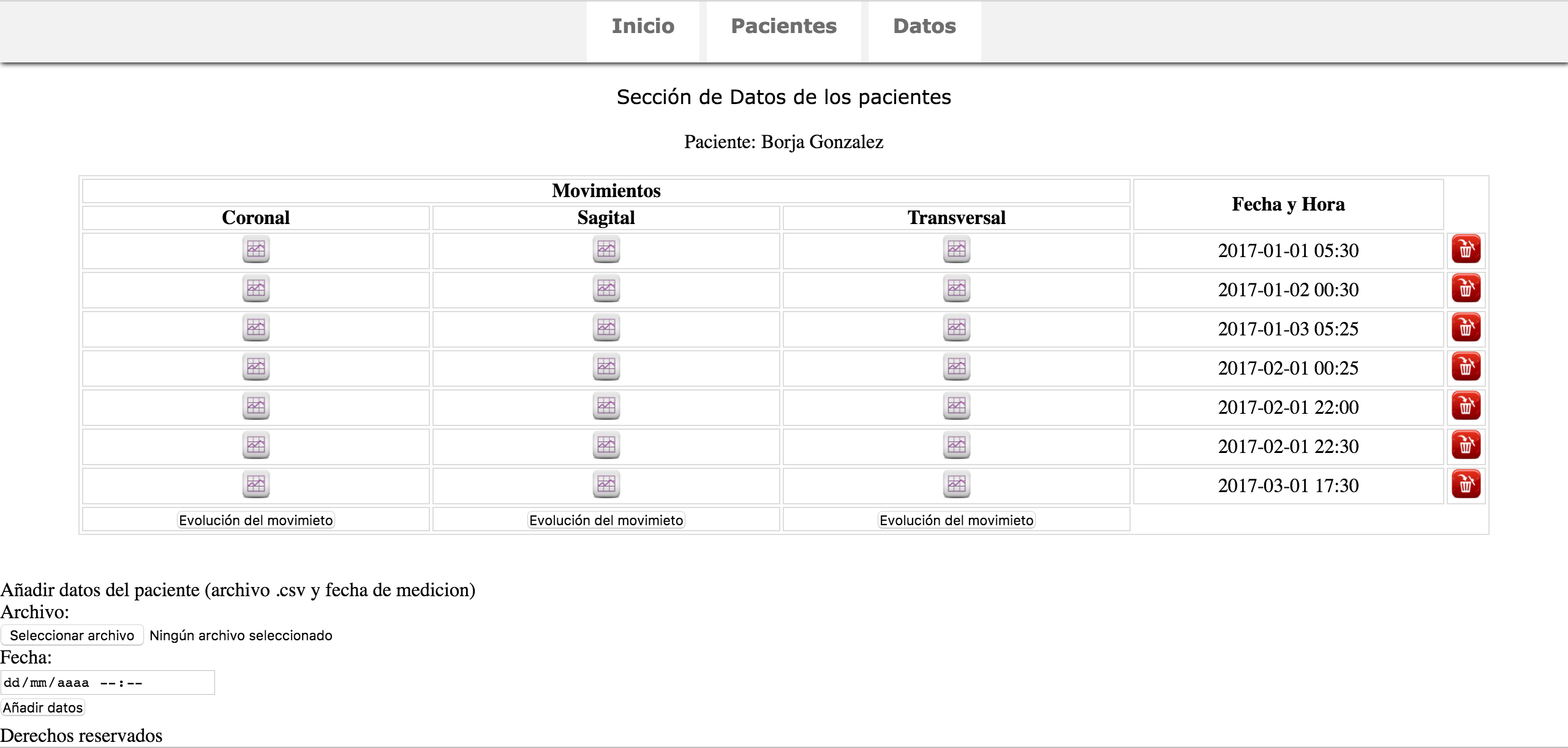
En la sección de inicio encontramos una breve descripción de cómo funciona la página web.

#### Sección de Pacientes

## 

En la sección de pacientes encontramos un listado de pacientes con el Nombre apellido un botón para acceder a los datos del paciente y otro botón para borrar al paciente. También existe la posibilidad de añadir pacientes, introduciendo su nombre, apellidos y sexo.

#### Sección de Datos

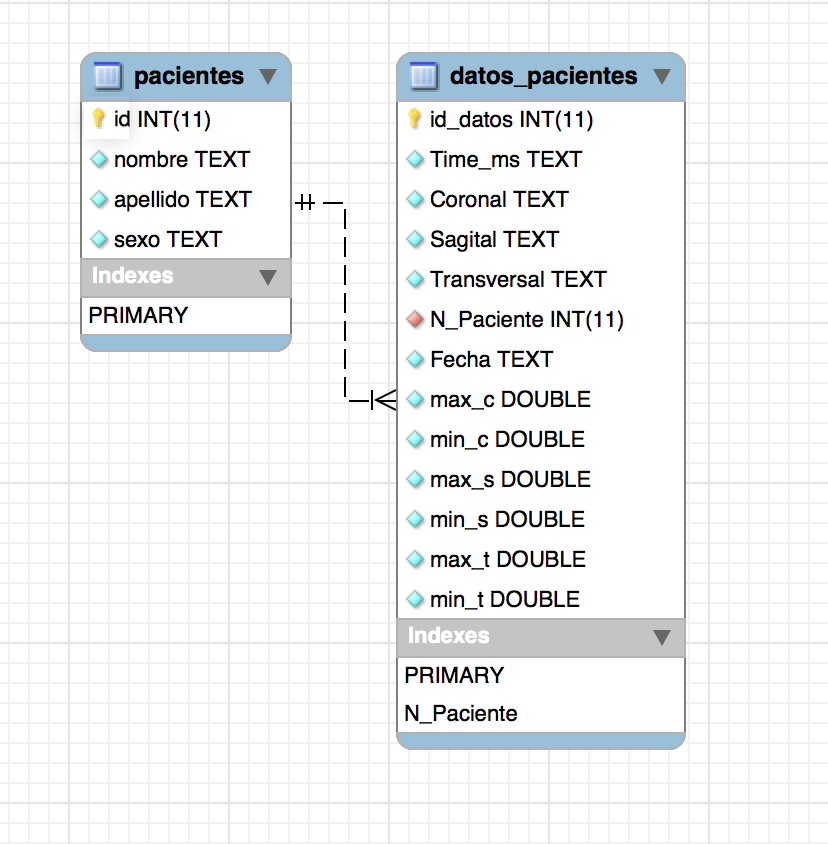


En la sección de datos tenemos un listado de conjuntos de movimientos (movimientos en los planos Coronal, Transversal y Sagital) con una fecha de medición asociada a cada conjunto. A demás es posible añadir un conjunto de datos con su fecha asociada y borrar un conjunto de datos.

### Aspecto de la Base de datos

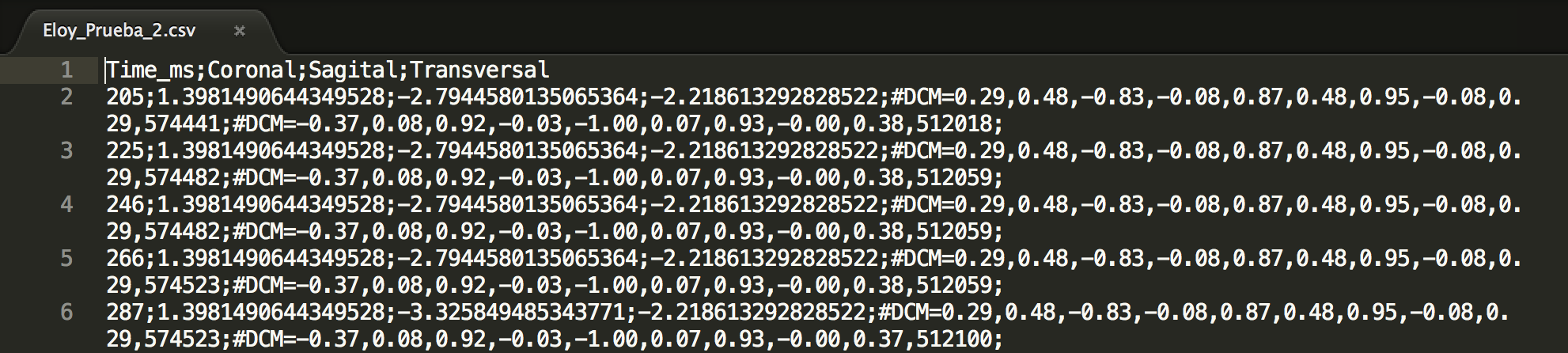
#### EER – Enhanced Entity-relationship model

Un modelo entidad-relación o diagrama entidad-relación (a veces denominado por sus siglas en inglés, E-R "Entity relationship"; en español DER: "Diagrama de Entidad-Relación") es una herramienta para el [modelado de datos](https://es.wikipedia.org/wiki/Modelo_de_datos" \o "Modelo de datos) que permite representar las entidades relevantes de un [sistema de información](https://es.wikipedia.org/wiki/Sistema_de_informaci%C3%B3n" \o "Sistema de información) así como sus interrelaciones y propiedades.



La base de datos está formada por dos tablas relacionadas (1…n) , lo que quiere decir que por cada paciente podrán existir varios sets de datos. En este tipo de relaciones siempre existe un Foreign Key, que es la clave que relaciona el set de datos con el paciente.

### Estructura del archivo CSV



En el archivo CSV podemos observar un documento de texto delimitado por un punto y coma (semicolon) que divide los datos en tiempo, movimiento coronal, movimiento sagital, movimiento transversal y datos adicionales que ayudan a calcular los previamente dichos. Éstos últimos datos no serán utilizados ya que no son necesarios.

## Especificación de la API

# Implementación

Pruebas

Resultados y conclusiones