

Escuela

Politécnica

Superior

Gala M. Ga

MINISTERIO DE DEFENSA | [Dirección de la compañía]

C:\Users\Gala\Documents\TFG\universidad.png

Implementación de una plataforma online segura para su uso en telemedicina

Autor:

Roberto André Vega Guardado

Tutores:

Sergio Orts Escolano

grado en ingeniería informática

Trabajo Fin de Grado

# Agradecimientos

**MOTIVACIÓN**

# Índice de contenido

1. Introducción
2. Objetivos

2.1 Objetivos específicos

2.2 Relación con asignaturas

1. Estado del arte
   1. Sistemas similares
2. Materiales y métodos
   1. Herramientas y tecnologías Software
   2. Herramientas Hardware
   3. Planificación
3. Especificación de requisitos
   1. Casos de uso
   2. Requisitos funcionales
   3. Requisitos no funcionales
4. Diseño y arquitectura
   1. Prototipos tempranos
   2. Diagrama de clases
   3. Diagrama de entidad relación
   4. Diseño final
   5. Arquitectura
5. Seguridad
   1. Servidor
   2. Cliente
6. Implementación
7. Conclusiones
8. Bibliografía
9. Anexo
10. **Introducción**

Las Tecnologías de la Información y Comunicación aplicadas a la atención en salud han impactado positivamente en el sector salud a nivel mundial en las últimas décadas. Cada vez es más notable el interés por la Telemedicina, una innovación que posiblemente jugará un papel importante en la evolución y transformación del sistema de salud en los próximos años. El empleo de telemedicina pueda solucionar problemas como el aumento de la demanda de servicios de salud o la necesidad de gestionar grandes cantidades de información (ehCOS, Telemedicina: una pieza clave en el futuro de la salud).

En la actualidad gracias a los avances tecnológicos, un gran número de personas tiene acceso a internet mediante un ordenador o un dispositivo móvil, lo que posibilita la telemedicina, pero con ello también aumenta el riesgo de que estas personas sufran ataques y sus datos personales se vean comprometidos. Esto no solo afecta al usuario sino también a las empresas, la ciberseguridad se ha convertido en uno de los mayores retos que afrontan las organizaciones, sea cual sea su tamaño. Una mala gestión de la seguridad puede tener tanto impacto económico, como afectar a la reputación y la confianza de socios y clientes.

Por eso es importante proporcionar un sistema que garantice la seguridad de los datos del usuario y evitar en máxima medida que se vean comprometidos ante cualquier ataque. En el caso de la aplicación web de consulta telemática el factor de la seguridad cobra más importancia ya que se trabaja con datos sensibles como pueden ser el historial clínico de los usuarios o mensajes entre médico y paciente.

1. **Objetivos**
   1. **Objetivos específicos**

El objetivo del trabajo es desarrollar un sistema de telemedicina seguro y de código abierto, que permita soportar la asistencia médica, independientemente de la distancia que separa a los que ofrecen el servicio, es decir, ofrecer una mejor comunicación entre médico y paciente sin la necesidad de que el paciente vaya a consulta personalmente.

Además, el sistema también dispondrá de otras funcionalidades útiles para los profesionales sanitarios y los usuarios como por ejemplo ver el historial clínico de un paciente y las correspondientes consultas anteriores, así como la posibilidad de crear nuevas. Tanto médicos como pacientes podrán crear una cita para acordar un día y una hora para una nueva consulta ya sea mediante videollamada o en persona si fuese necesario. También el personal sanitario podrá comunicarse con los pacientes mediante mensajes.

Al ser la seguridad uno de los principales objetivos, para la implementación y el desarrollo de este sistema se tendrán en cuenta todos los aspectos relacionadas con este aspecto. A la hora de acceder al sistema se dispondrá de un doble factor de autenticación, mediante contraseña y validación de un código enviado por correo electrónico. Con respecto a la persistencia de datos, se cifrará todo dato relevante mediante algoritmos criptográficos antes de almacenarlo en la base de datos para garantizar la seguridad de los mismos. Y en cuanto a la comunicación y transferencia de los datos, el sistema dispondrá del protocolo HTTPS para el tráfico de información sensible.

* 1. **Relación con asignaturas**

En este apartado se describirá la relación y el conocimiento que han aportado algunas de las asignaturas cursadas a lo largo de la carrera, al proyecto.

**Seguridad en el diseño software (SDS)**

En esta asignatura es donde se dan los principales conocimientos de seguridad informática aplicados en el proyecto: criptografía, cifrados, funciones hash, seguridad a nivel de transporte, malware, ataques…

**Aplicaciones distribuidas en internet (ADI)**

En ADI es donde se enseña el desarrollo de APIs REST utilizando el lenguaje JavaScript. Además, se aprende acerca sobre el desarrollo en la parte cliente con JavaScript estándar y con frameworks. Para el proyecto se han utilizado las tecnologías usadas en esta asignatura tanto para el API como para la parte del cliente.

**Diseño de bases de datos (DBD)**

Los principales conocimientos de bases de datos aplicados en el proyecto se obtuvieron de esta asignatura. Poder diseñar un esquema entidad relación según los requisitos del sistema, creación de una base de datos desde cero…

**Análisis y especificación de sistemas software (AESS)**

En esta asignatura se aprende a realizar los diferentes diagramas del sistema, ya sean de clases, de casos de uso, de secuencia… Otro aspecto de la asignatura que se ha aplicado al proyecto es identificar y diferenciar los requisitos de la aplicación a desarrollar.

1. **Estado del arte**

Hoy en día el uso de un dispositivo con acceso a internet es habitual dentro de la vida diaria de las personas. Cada vez es más habitual que las personas busquen comodidades o alternativas a través de la tecnología para facilitar hábitos de la vida cotidiana.

Según el informe del 2018 que presentan We Are Social y Hootsuite se destaca que Internet ha superado la cifra de los 4.000 millones de usuarios, lo que supone que más de la mitad de la población está conectada.

******En concreto en España, los usuarios de Internet han aumentado en 2 millones, lo que supone ya el 85% de la población. Según el informe Sociedad Digital en España 2017 elaborado por la Fundación Telefónica detalla que el 98% de los jóvenes se conecta a diario a Internet, un porcentaje que en el caso de los adultos de entre 65 y 74 años es del 43,7% (Expansión). A este respecto, añade que las personas mayores se están incorporando “con rapidez” a la sociedad digital, ya que su uso de Internet aumentó un 26% el último año (La Vanguardia).

Ilustración 1 : Tasa de uso de internet por países (We Are Social).

Todo esto supone una facilidad para que la telemedicina sea una alternativa factible en el ámbito de la salud, ya que es un recurso tecnológico que posibilita la optimización de los servicios de atención en salud, ahorrando tiempo y dinero y facilitando el acceso a zonas distantes para tener atención de especialistas. Factores como la alta dispersión poblacional, las distancias y la escasez de especialistas en todas las áreas de la medicina dan pie a que la telemedicina hoy en día pueda desempeñar un papel fundamental en lo que concierne a la mejora sostenible de la salud de las comunidades, a nivel global.

Es una herramienta más para el buen desempeño científico del personal de la salud, que no solucionará todos los problemas existentes en el sector sanitario y que en ningún caso pretende sustituir el criterio del médico tratante, pero que con la evolución de las nuevas tecnologías favorece la expansión y aplicación de los conceptos de globalidad e interoperabilidad en las organizaciones de salud, permitiendo e impulsando así entornos de organización y trabajo que van más allá de la aplicación de la Telemedicina para la cobertura asistencial en zonas geográficamente remotas.

En la Tabla 1 se puede observar un resumen de las ventajas y los inconvenientes de la telemedicina.

|  |  |
| --- | --- |
| Ventajas | Inconvenientes |
| Acorta las distancias entre especialista y paciente. | Escasa relación médico-paciente |
| Reducción de costes | Puede ser impersonal |
| Última tecnología al alcance de todos | La calidad de la información puede ser deficiente |
| Acceso a sitios remotos | Necesidad de capacitaciones adicionales |
| Educación continua más asequible | Calidad de la información de la salud incierta |
| Mejor comunicación entre los profesionales de la salud | La confidencialidad se puede ver comprometida |

Tabla 1: Ventajas e inconvenientes de la telemedicina

El mercado de la telemedicina, el llamado e-Healt y m-Healt, generó un volumen de negocio de más de 6000 millones de euros en 2015, según Jaime Hortelano, el consejero delegado de la consultora española CMC. La expansión de los dispositivos móviles ha establecido ya una base tecnológica para poder desarrollar rápidamente la telemedicina que muy pronto será un valor añadido de los servicios que ofrecen las aseguradoras (ScienceDirect, Telemedicina estado actual).

Dentro del mercado de las aplicación web y móviles, voy a analizar algunas de las aplicaciones de telemedicina más completas e interesantes.

* 1. **Sistemas similares**
     1. **Médicos y pacientes**

Médicos y pacientes es una plataforma que permite a los usuarios acceder de forma segura y confidencial a su historial clínico y contactar con el medico telemáticamente entre otras funcionalidades.

Entre sus características hay que destacar:

* Login con clave o certificado digital
* Consulta de tarjeta sanitaria
* Consulta del centro de salud (mapa, números, etc.)
* Hospital de referencia
* Médico y enfermero de cabecera
* Citas electrónicas
* Ver historial clínico
* Datos de facturación farmacéutica
* Consulta online de determinadas dudas sobre su asistencia sanitaria (responde el médico)
* Ver información del paciente
  + 1. **Policlínica IUMET**

El Instituto Universitario de Medicina Telemática IUMET surge para dar a los Centros de Reconocimiento de Conductores (CRC). Entre sus servicios encontramos:

* Realizar consultas a los médicos especialistas de manera instantánea sobre las exploraciones psicofísicas a los conductores que acuden al Centro
* Obtener apoyo formativo que amplíe y afiance los conocimientos y habilidades de los facultativos del Centro de Reconocimiento
* Promover la investigación en el ámbito de la seguridad vial, a través de la coordinación y análisis de datos a gran escala
* Asesoramiento en adaptaciones para la conducción
* Teleoftalmología para Centros de Reconocimiento de Conductores
* Dispositivos de telecomunicación: dependiendo del tipo de telemedicina. el equipamiento puede estar compuesto por terminales telefónicos, estaciones de videoconferencia (como tablets con webcam incorporada u ordenadores con webcam externa)
  + 1. **Vida**

Vida es una aplicación para realizar consultas médicas a través de una videollamada directamente en la aplicación, en la que médicos experimentados y colegiados te ofrecen ayuda médica y si es necesario, pueden prescribir una receta.

Entre sus características tenemos:

* Video consultas para:
  + Preocupaciones generales de salud
  + Alergias
  + Asma
  + Infecciones sinusales
  + Herpes labial
  + Resfriados
  + Infección del tracto urinario
* Reservar citas para la videoconsulta
* La primera consulta es gratis, después cada una tiene un coste de 10€
* El médico podrá recetar
* Es el doctor el que inicia la llamada
* Las consultas no tienen límite de tiempo
  + 1. **Sanitas**

La plataforma de sanitas dedicada a la telemedicina se llama BLUA, con ella puedes conectar los médicos y realizar consultas online. Mediante la videoconsulta a través de un portátil, tablet o móvil ofrecen una atención médica personalizada.

Entre los servicios que ofrecen se encuentran:

* Cobertura de farmacia: reembolso del 50% de los gastos de farmacia.
* Cobertura en EE.UU
* Cobertura de reembolso 50.000: permiten acceder a un especialista que aunque no esté en su cuadro médico y te reembolsan el 80% de los gastos médicos.
* Analítica a domicilio: reembolso de los gastos del personal de laboratorio a domicilio para mayor comodidad.
* Envío de medicamentos a domicilio.
* Asesor medico
* Programa de psicología y de nutrición
  + 1. **Atrys**

Atrys es una compañía biomédica dedicada a la prestación de servicios diagnósticos y tratamientos médicos de excelencia, cuyo objetivo es facilitar la terapia individualizada de los pacientes, así como el desarrollo de nuevas modalidades terapéuticas y herramientas diagnósticas.

En el apartado de la telemedicina sus servicios en las especialidades médicas son:

* Radiología
* Cardiología
* Oftalmología
* Dermatología.
  + 1. **HealthTap**

HealthTap es una aplicación gratuita tanto para web como para móvil y algunas de sus características son las siguientes:

* Consultas virtuales con acceso las 24h a través de mensajes de voz, de texto o video.
* Posibilidad de que un especialista recete tratamientos al paciente
* recordatorios, listados
* Boletines personificados por doctores.
* Asesoramiento de especialistas

1. **Materiales y métodos**
   1. **Herramientas y tecnologías software**

A continuación, se van enumerar y describir brevemente las distintas herramientas software usadas para el desarrollo del sistema.

* + 1. **Documentación**

Para realizar la documentación del proyecto se ha utilizado principalmente un programa:

* **Microsoft Word:** programa que ofrece creación de documentos y procesamiento de textos entre otras muchas características. Se ha usado para elaborar la memoria en su totalidad.
  + 1. **Diseño**

En el apartado de las herramientas de diseño que han intervenido en el desarrollo del sistema y de la documentación tenemos:

* **Draw.io:** es una aplicación de código abierto para la elaboración de diagramas de aplicaciones. Esta herramienta se ha utilizado para la mayoría de diagramas del sistema, es decir, los diagramas de casos de uso, el diagrama de clases y para los prototipos tempranos.
  + 1. **Desarrollo**

Entre las herramientas comunes para el desarrollo del sistema, tanto del API, como de la parte del cliente web y de la base de datos, están las siguientes:

* **Git:** es un software de control de versiones para proyectos, que permite llevar un registro de los cambios realizados y así poder coordinar el trabajo con otras personas.
* **GitHub:** es una plataforma para alojar proyectos usando el sistema de control de versiones Git.

Para la parte del API y de la gestión de la base de datos se ha hecho uso de las siguientes tecnologías:

* **Visual Studio Code:** es un editor de código fuente gratuito y de código abierto desarrollado por Microsoft. Incluye soporte para la depuración, control integrado de Git , resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código (Wikipedia).
* **NodeJs:** es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor que ejecuta código JavaScript fuera del navegador. Está basado en el lenguaje de programación ECMAScript, asíncrono y orientado a eventos, lo que lo hace útil para la creación de servidores web (Wikipedia).
* **JavaScript:** lenguaje de programación interpretado orientado a objetos y basado en prototipos, el cual se utiliza tanto en la parte del servidor como en la del cliente.
* **ExpressJS:** es un framework para NodeJS de código abierto, proporciona métodos de utilidad para HTTP y middleware.
* **Postman:** es un programa multiplataforma utilizado para el desarrollo de APIs, ya que permite ejecutar peticiones, hacer pruebas, depurar, crear pruebas automatizadas y documentar un API. En el proyecto se ha utilizado también para documentar las peticiones junto con sus respuestas para tener la información en línea.
* **MySQL:** es un sistema de gestión de bases de datos relacional, gracias a su rendimiento, confiabilidad y facilidad de uso se ha convertido en el principal sistema de bases de datos en el mundo.
* **MySQLWorkbench:** es una herramienta visual unificada para arquitectos de bases de datos, desarrolladores y DBA. MySQL Workbench proporciona modelado de datos, desarrollo de SQL y herramientas integrales de administración para la configuración del servidor, administración de usuarios, respaldo y mucho más (MySQL).

Para la parte del cliente se han usado las siguientes tecnologías comunes a la parte del API de Visual Studio como editor de código y JavaScript como lenguaje. El resto de herramientas son:

* **ReactJS:** es una librearía de JavaScript para crear interfaces de usuario interactivas basada en componentes que encapsulan un comportamiento, una vista y un estado, lo que proporciona facilidad de mantenimiento, depuración y escalabilidad.
* **Webpack:** es un paquete de módulos para JavaScript de código abierto, su principal objetivo es agrupar los archivos de JavaScript con sus dependencias y genera archivos estáticos para su uso en un navegador, pero también es capaz de transformar, agrupar o empaquetar.

* 1. **Herramientas Hardware**

Para el correcto funcionamiento del sistema no es necesario hardware adicional aparte del servidor. La primera opción es tener alojado el API en un PaaS (*Plataform as a service*), lo que permite la automatización del ciclo de vida, configuración, despliegue y escalado de software, sin la complejidad de comprar, crear o administrar toda la infraestructura necesaria. La segunda opción es montar esa infraestructura por nuestra cuenta, en ese caso, en un entorno de producción un servidor node debería de tener las siguientes características:

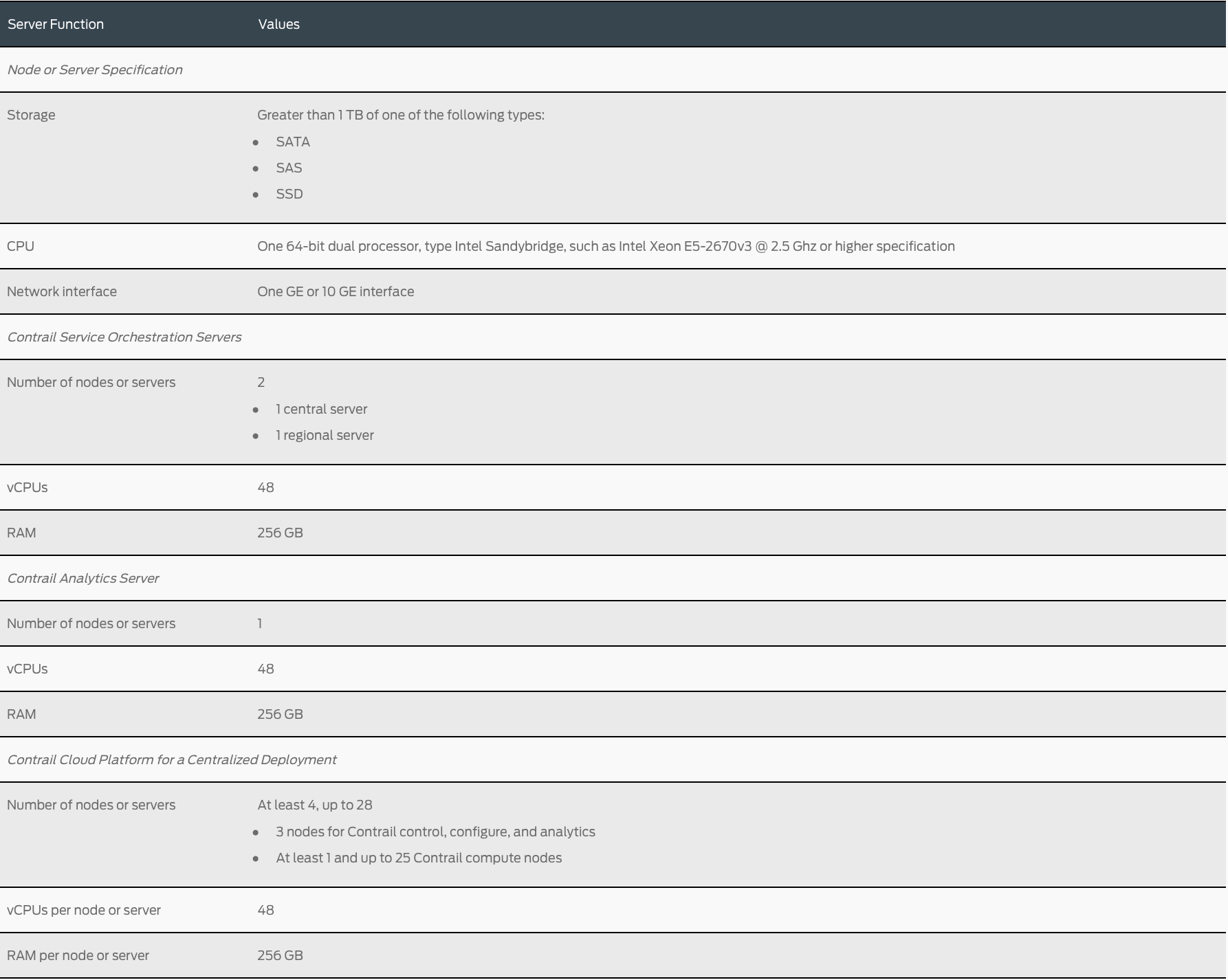


Tabla 2: Entorno de producción (juniper.net)

* 1. **Planificación**

Para la planificación del proyecto, se optó por dividir el proyecto en tres partes diferenciadas:

* API
* Cliente
* Documentación

A continuación se explicara cómo se ha gestionado ésta planificación y qué herramientas se han utilizado para ello.

* + 1. **Trello**

Para llevar la gestión de estas tres partes se ha utilizado la herramienta **Trello**, que resumidas cuentas es una plataforma para la administración de proyectos. En ella se ha creado un tablero del proyecto global con las tres partes, cada parte se diferencia con una etiqueta con un color diferente. Tenemos que las tareas etiquetadas con el color azul pertenecen a la parte del API, las de color rojo a la parte del cliente y las de color verde para la parte de la documentación.

En el tablero (*Ilustración 2*) se optó por poner cuatro columnas:

* Listado de tareas: aquí es donde están las tareas pendientes.
* En desarrollo: aparecen las tarjetas con las tareas que se están desarrollando y están en proceso.
* Pruebas: es la columna donde aparecen las tarjetas con las tareas que se han terminado pero no se han probado en profundidad.
* Terminado: es la columna donde están las tareas terminadas.

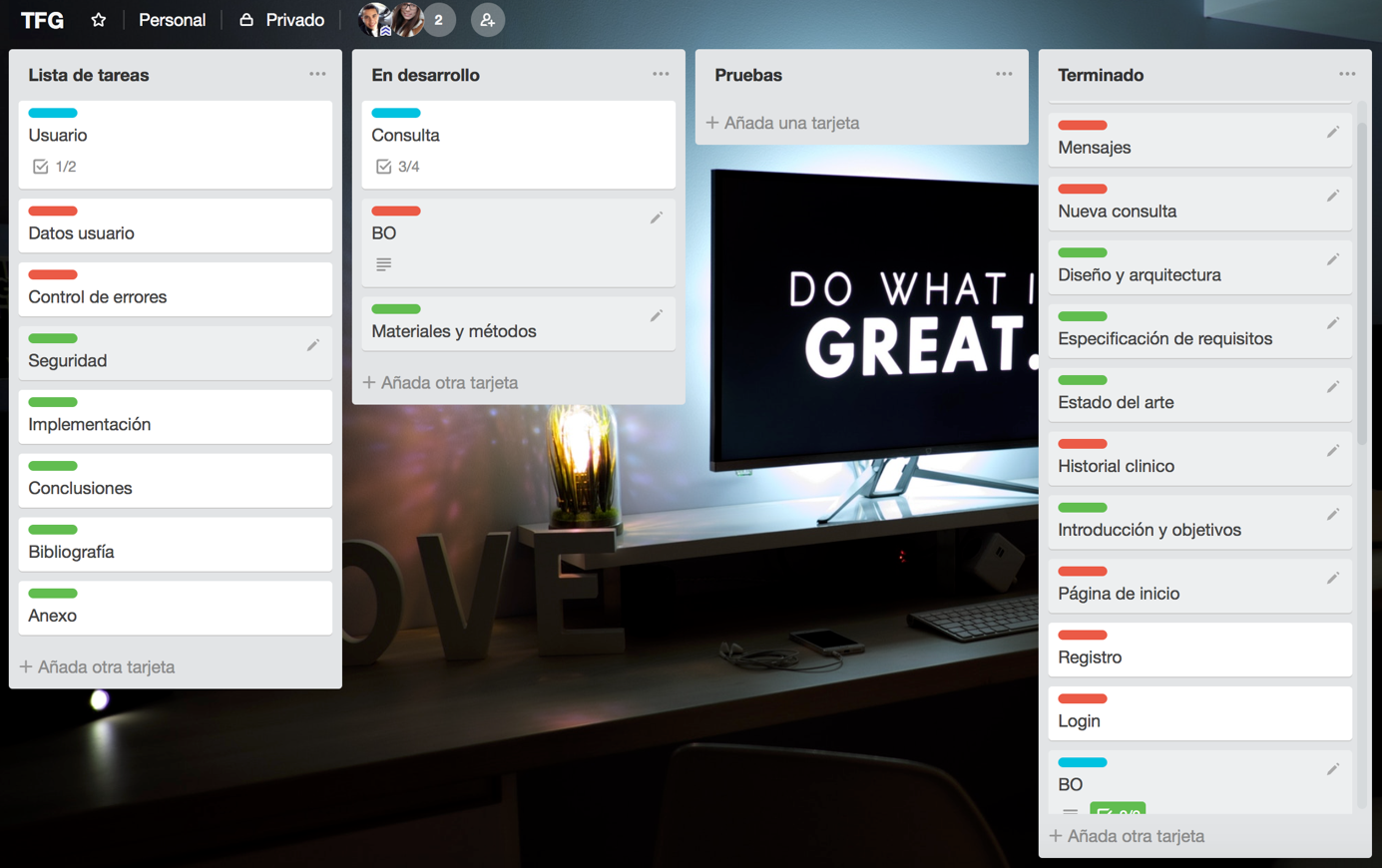
****

Ilustración 2: Tablero de Trello con el estado del proyecto

Cada parte del proyecto a su vez, se ha dividido en elementos más pequeños, estos elementos son los llamados “historias de usuario”. Estas historias se han dividido en tareas mas pequeñas dentro de cada tarjeta utilizando los “checklist” que proporciona Trello.

En la *Ilustración 3* podemos observar que en la columna de en desarrollo coinciden tareas de las 3 partes, debido a que se ha intentado desarrollar las tres partes a la vez, aunque al principio se le diera más prioridad a desarrollar el API, por eso en la imagen la mayoría de tareas asignadas al API están en la columna terminado.

* + 1. **Diagrama de Gant**

Para poder ver una planificación más visual del proyecto se ha utilizado un diagrama de Gantt, es una herramienta gráfica cuyo objetivo es planificar y exponer el tiempo de dedicación previsto para las diferentes tareas a lo largo de un tiempo total determinado.

Para la realización del diagrama se ha usado la web de Monday.com, la cual permite crear tableros de muchas temáticas y observar el progreso con distintos tipos de vistas. Para el proyecto se optó por crear tres grupos: servidor, cliente y memoria.

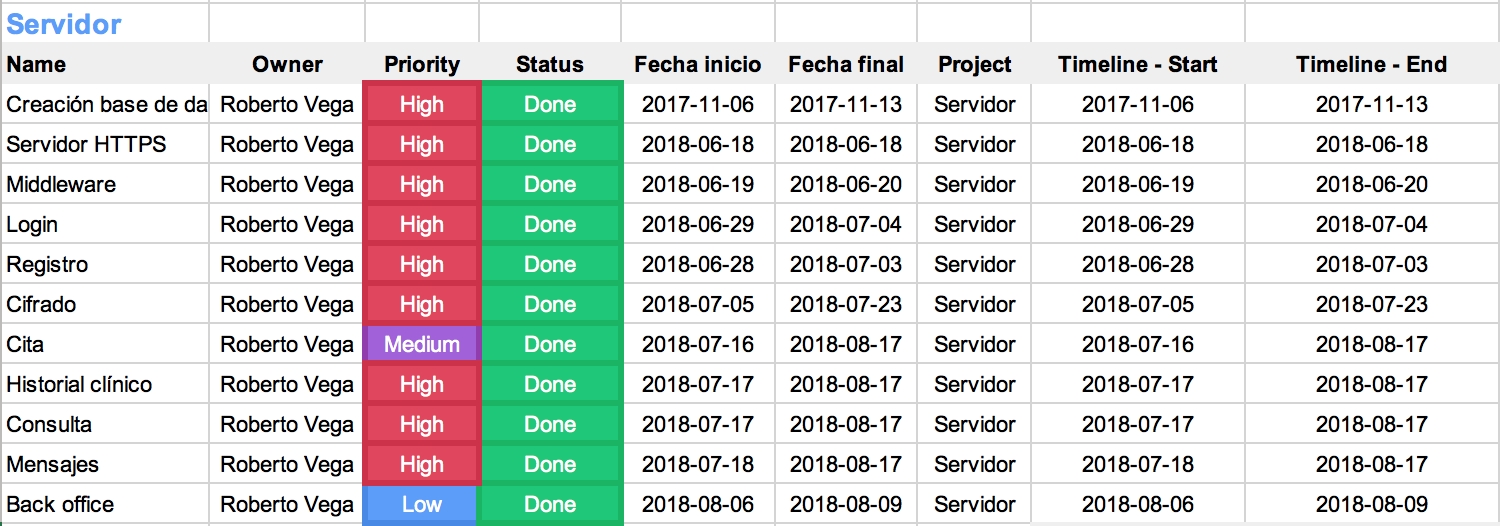


Tabla 3: Grupo de tareas del servidor

En la *tabla 3* podemos observar el grupo de tareas del servidor, en ella aparecen el nombre de las tareas, el que las ha realizado, la prioridad de cada tarea y la fecha de inicio y final de cada una.

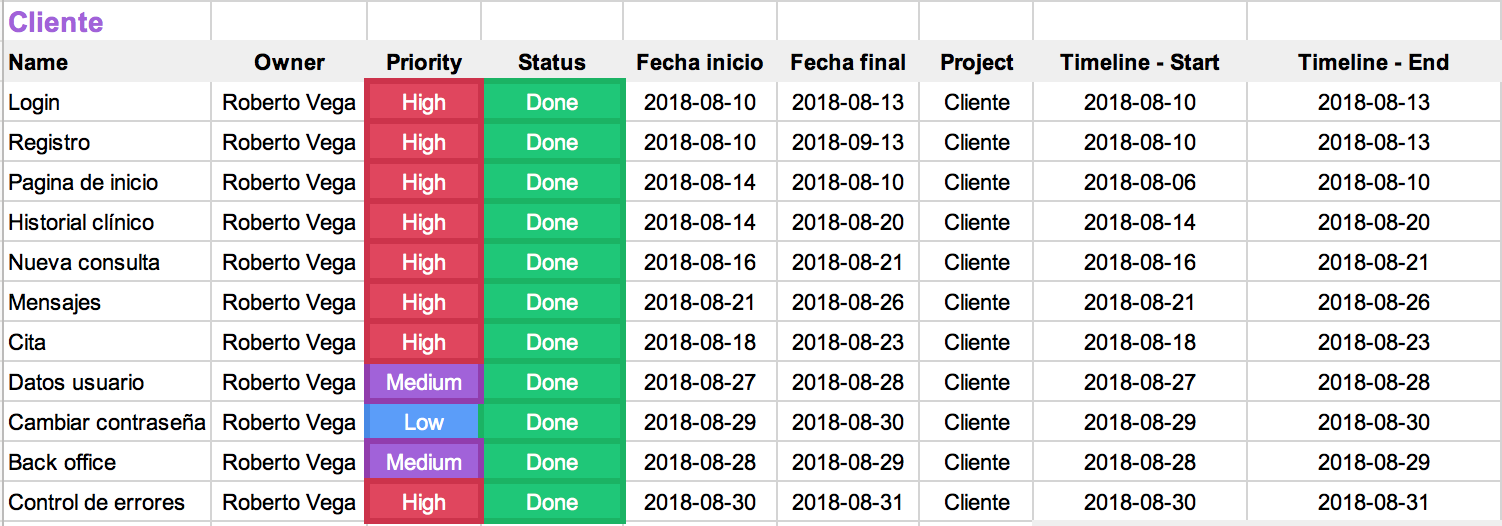


Tabla 4: Grupo de tareas del cliente

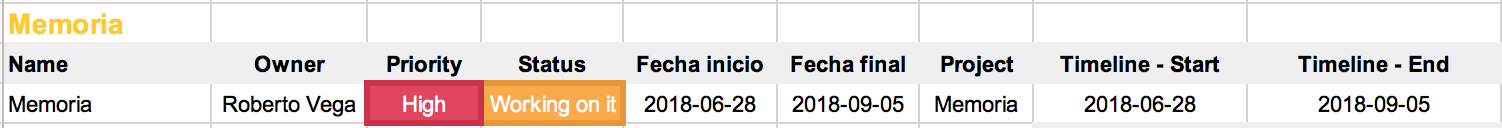


Tabla 5: Grupo de tareas memoria

En la *tabla 4* tenemos las tareas de la parte del servidor con la misma información que en la del servidor. La diferencia está en la *tabla 5* que, como se puede observar, se ha optado por poner solo una tarea que generaliza todo el apartado de la documentación.

Finalmente en la *ilustración 3* podemos observar el progreso general de las tareas de los tres grupos.

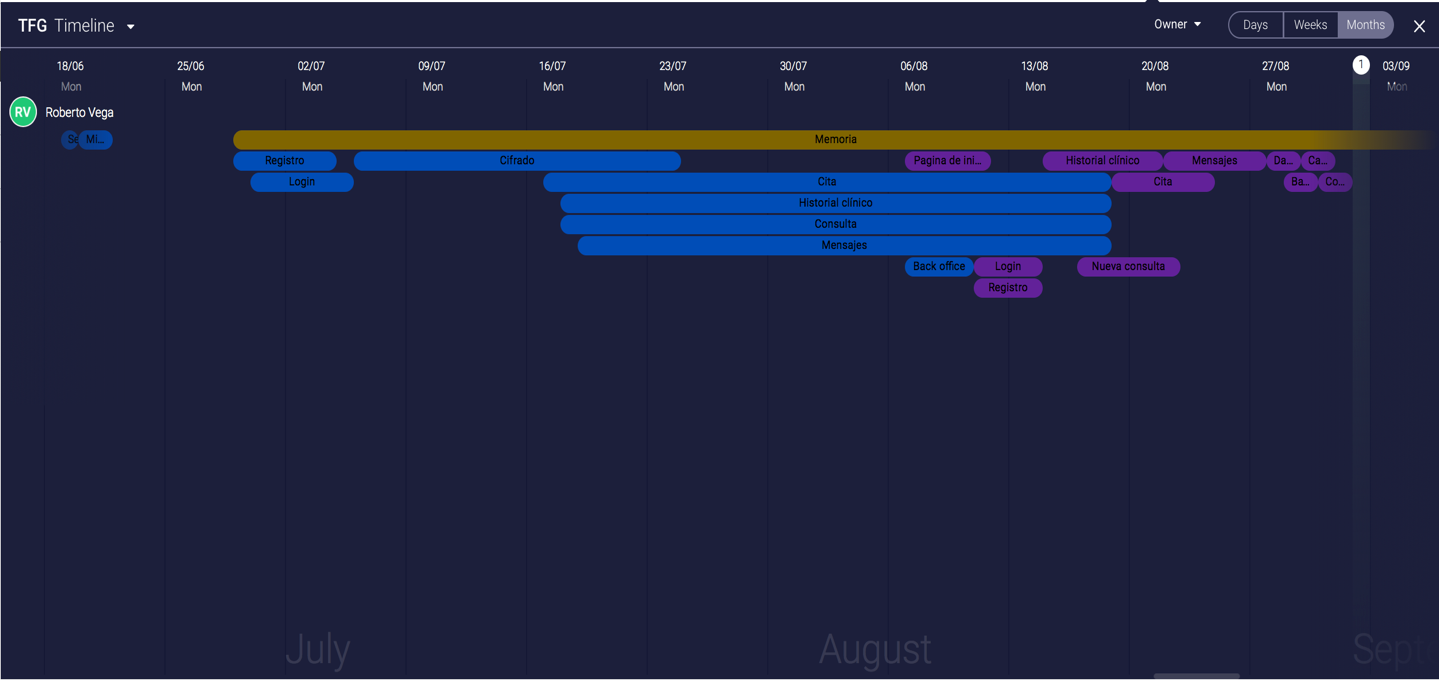


Ilustración 3: Timeline

Mirando la imagen podemos observar la secuencia del desarrollo del proyecto que se ha seguido. Al no haber seguido un desarrollo continuo, en el diagrama falta la primera tarea que es la de “creación de base de datos”, que como podemos ver en la *tabla 3* se hizo al principio, mucho antes que el resto de tareas, es por eso que no aparece en el diagrama de la *ilustración 3,* donde solo aparece un rango de fechas. Las tareas aparecen según el color del grupo en el que estén: las azules pertenecen al servidor, las violeta a la parte del cliente y por ultimo la tarea general de la documentación que aparece en amarillo.

1. **Especificación de requisitos**

El análisis de requisitos sirve para describir los requerimientos del sistema, es decir, los servicios que ha de ofrecer el sistema, la utilidad que van a tener esos servicios para el usuario y las restricciones asociadas a su funcionamiento. El principal objetivo es facilitar la comunicación entre cliente, usuario y desarrollador.

Ésta especificación es importante para el futuro éxito del sistema, ya que unos requisitos mal formulados dan como resultado un producto deficiente y o de poca calidad. El coste de arreglar un producto deficiente es mucho mayor que el coste de un buen análisis y especificación de requisitos.

* 1. **Casos de uso**

Los diagramas de casos de uso muestran una descripción visual de los pasos que han de llevarse a cabo para realizar un proceso del sistema. El objetivo principal de los casos de uso es describir la manera en que se usará el sistema, es decir, detallar las funcionalidades esenciales y así mejorar la compresión del sistema para desarrolladores, usuarios finales y expertos del dominio.

Un caso de uso contiene una descripción textual de todas las maneras que los actores previstos podrían trabajar con el software o el sistema, es decir, muestran los pasos que el actor debe seguir para poder realizar con éxito una tarea.

En nuestro caso en el sistema intervienen tres actores. El médico que, aparte de las funciones que comparte con el paciente, puede ver el historial clínico con las consultas asociadas y crear nuevas consultas. El paciente puede ver el historial de citas, crear una nueva cita, ver sus datos del historial clínico y enviar mensajes a otro usuario. Y finalmente el administrador tiene las funciones de validar un nuevo medico registrado y añadir el historial clínico de un nuevo paciente registrado.

Los diagramas de casos de uso del sistema son los siguientes:

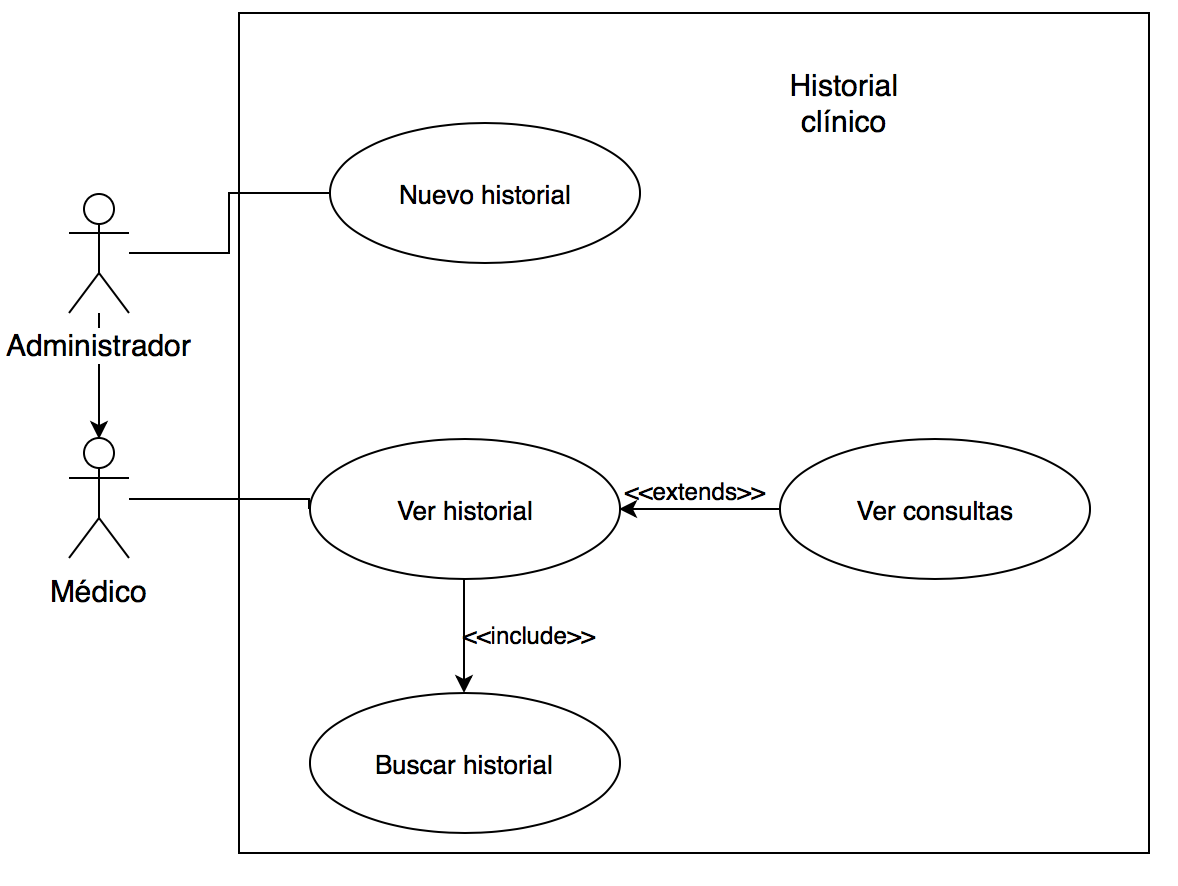


Ilustración 4: Diagrama de caso de uso 1

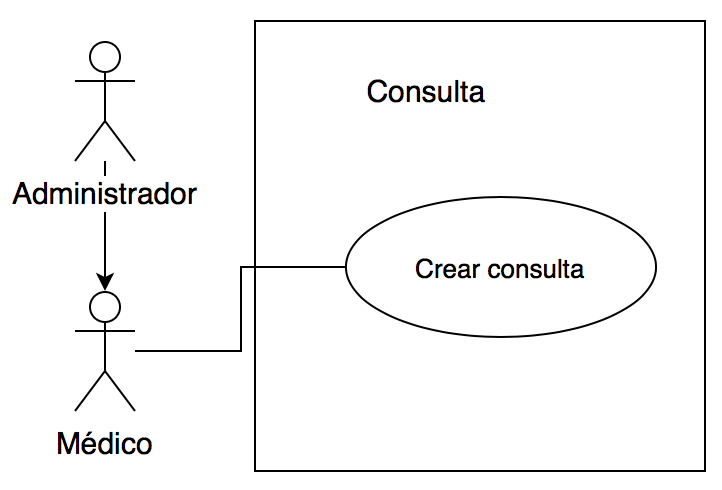


Ilustración 5: Diagrama de caso de uso 2

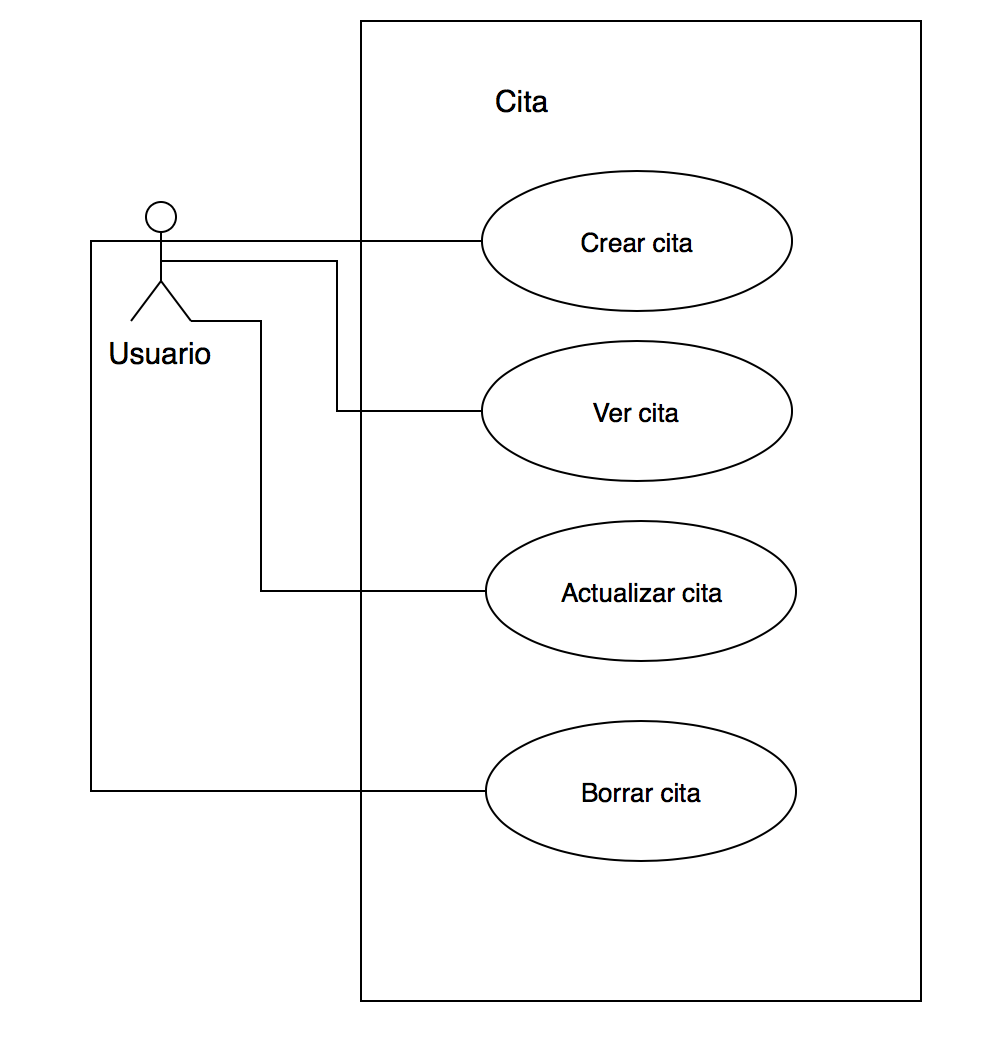


Ilustración 6: Diagrama de caso de uso 3

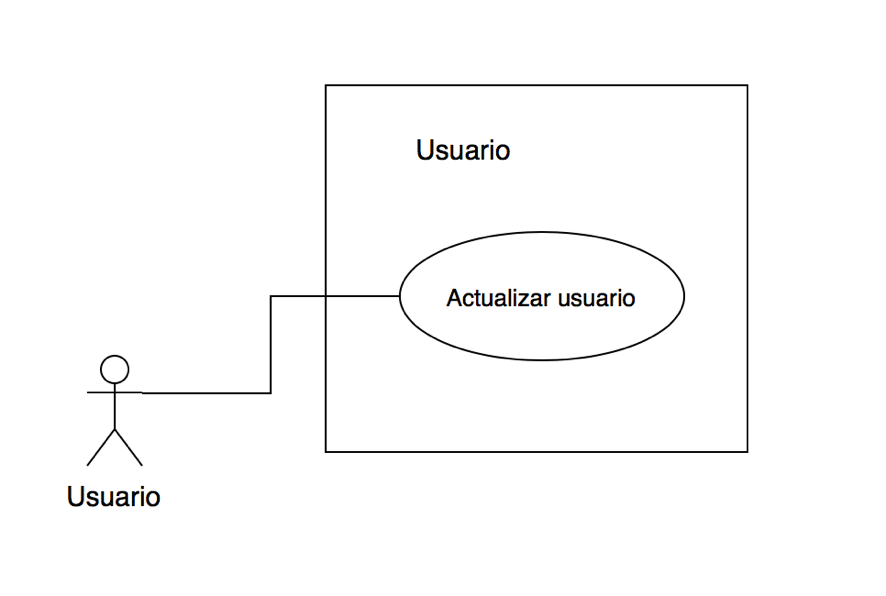


Ilustración 7: Diagrama de caso de uso 4

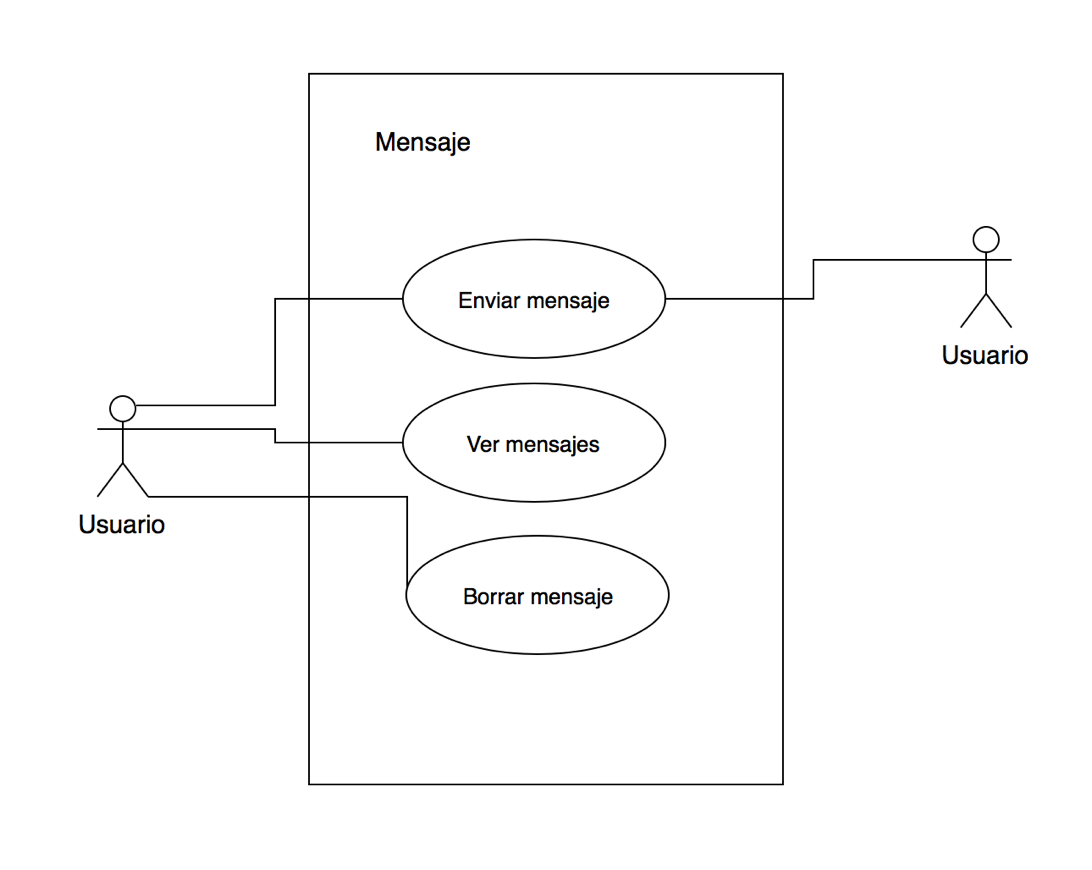


Ilustración 8: Diagrama de caso de uso 5

Los detalles de cada caso de uso se describen en su correspondiente tabla. En cada ellas podemos encontrar el título del caso de uso, el actor que realiza la tarea y el curso normal de la tarea especificando si algún paso de la secuencia se puede hacer de forma alternativa.

En el sistema se pueden diferenciar dos actores que son los que realizan las principales tareas de la aplicación, el médico y el paciente.

|  |  |
| --- | --- |
| Caso de uso | Ver el historial de un paciente |
| Actor | Médico |
| Curso normal | **Curso alternativo** |
| 1. El médico inicia sesión en el sistema. | 1.1 Si el médico no está registrado, se le informará que debe de hacerlo. |
| 2. El médico va al apartado de historial. |  |
| 3. El médico buscará la paciente según su SIP. |  |
| 4. El médico podrá ver el historial clínico y las consultas de ese paciente |  |

Tabla 6: Caso de uso 1

|  |  |
| --- | --- |
| Caso de uso | Nueva consulta |
| Actor | Médico |
| Curso normal | **Curso alternativo** |
| 1. El médico inicia sesión en el sistema. | 1.1 Si el médico no está registrado, se le informará que debe de hacerlo. |
| 2. El médico va al apartado del historial y luego al de nueva consulta. |  |
| 3. El médico introducirá los datos necesarios para la nueva consulta. |  |
| 4. El médico guardará la nueva consulta |  |

Tabla 7: Caso de uso 2

|  |  |
| --- | --- |
| Caso de uso | Citas |
| Actor | Médico y paciente |
| Curso normal | **Curso alternativo** |
| 1. El médico/paciente inicia sesión en el sistema. | 1.1 Si el médico/paciente no está registrado, se le informará que debe de hacerlo. |
| 2. El médico/paciente va al apartado de citas. |  |
| 3. El médico/paciente podrá ver las citas con la opción de filtrarlas según el tipo (videollamada o presencial), crear una nueva cita, actualizar y borrar una cita existente. |  |

Tabla 8: Caso de uso 3

|  |  |
| --- | --- |
| Caso de uso | Actualizar datos |
| Actor | Médico y paciente |
| Curso normal | **Curso alternativo** |
| 1. El médico/paciente inicia sesión en el sistema. | 1.1 Si el médico/paciente no está registrado, se le informará que debe de hacerlo. |
| 2. El médico/paciente va al apartado de mi cuenta y editar perfil. |  |
| 3. El médico/paciente tendrá la opción de cambiar su contraseña o su nombre y apellidos |  |

Tabla 9: Caso de uso 4

|  |  |
| --- | --- |
| Caso de uso | Mensajes |
| Actor | Médico y paciente |
| Curso normal | **Curso alternativo** |
| 1. El médico/paciente inicia sesión en el sistema. | 1.1 Si el médico/paciente no está registrado, se le informará que debe de hacerlo. |
| 2. El médico/paciente va al apartado de mensajes. |  |
| 3. El médico/paciente podrá ver sus mensajes recibidos o enviados, enviar uno nuevo o borrar alguno ya existente. |  |

Tabla 10: Caso de uso 5

* 1. **Requisitos funcionales**

Los requerimientos funcionales son declaraciones de los servicios o funciones que proveerá el sistema, de la manera en que éste reaccionará a entradas particulares. Una función es descrita como un conjunto de entradas, comportamientos y salidas.

A continuación, se muestran los requisitos funcionales del sistema mostrados en tablas, en ellas se muestra el nombre del requisito, una breve descripción y la prioridad del mismo.

|  |  |
| --- | --- |
| Nombre | Ver historial clínico |
| Descripción | Un médico podrá ver el historial clínico introduciendo la SIP de un paciente y con ello todas las consultas anteriores. |
| Prioridad | Alta |

Tabla 11: Requisito funcional 1

|  |  |
| --- | --- |
| Nombre | Crear una nueva consulta |
| Descripción | Un médico podrá crear una nueva consulta, rellenando un formulario con los datos necesarios, asociada al historial clínico de un paciente. |
| Prioridad | Alta |

Tabla 12: Requisito funcional 2

|  |  |
| --- | --- |
| Nombre | Pedir cita |
| Descripción | Un paciente o un médico podrá solicitar una cita en un día y una hora concretos. |
| Prioridad | Alta |

Tabla 13: Requisito funcional 3

|  |  |
| --- | --- |
| Nombre | Ver cita |
| Descripción | Un paciente o un médico podrá ver sus citas con la posibilidad de filtrarlas por el tipo de cita, ya sea videollamada o presencial. |
| Prioridad | Media |

Tabla 14: Requisito funcional 4

|  |  |
| --- | --- |
| Nombre | Actualizar cita |
| Descripción | Un paciente o un médico podrá actualizar la hora y el día de una cita ya existente. |
| Prioridad | Media |

Tabla 15: Requisito funcional 5

|  |  |
| --- | --- |
| Nombre | Borrar cita |
| Descripción | Un paciente o un médico podrá borrar una cita ya existente. |
| Prioridad | Media |

Tabla 16: Requisito funcional 6

|  |  |
| --- | --- |
| Nombre | Actualizar datos personales |
| Descripción | Un paciente o un médico podrá cambiar su contraseña o su nombre y apellidos. |
| Prioridad | Baja |

Tabla 17: Requisito funcional 7

|  |  |
| --- | --- |
| Nombre | Enviar un mensaje |
| Descripción | Un paciente o un médico podrán enviar un mensaje. |
| Prioridad | Alta |

Tabla 18: Requisito funcional 8

|  |  |
| --- | --- |
| Nombre | Ver mensajes |
| Descripción | Un paciente o un médico podrán ver sus mensajes recibidos o los enviados. |
| Prioridad | Alta |

Tabla 19: Requisito funcional 9

|  |  |
| --- | --- |
| Nombre | Borrar un mensaje |
| Descripción | Un paciente o un médico podrán borrar sus mensajes enviados. |
| Prioridad | Alta |

Tabla 20: Requisito funcional 10

* 1. **Requisitos no funcionales**

Son aquellos requerimientos que no se refieren directamente a las funciones específicas que entrega el sistema, sino a las propiedades emergentes de éste como la fiabilidad, la respuesta en el tiempo y la capacidad de almacenamiento. De forma alternativa, definen las restricciones del sistema como la capacidad de los dispositivos de entrada/salida y la representación de datos que se utiliza en la interface del sistema.

Los requisitos no funcionales también se muestran en tablas donde aparece el nombre, descripción y prioridad de cada requisito.

|  |  |
| --- | --- |
| Nombre | Seguridad |
| Descripción | * La aplicación al trabajar con datos muy sensibles debe garantizar la protección e integridad de esos datos. * El sistema debe trabajar con las versiones más actualizadas de algoritmos de cifrado para asegurarse que no tengan vulnerabilidades. |
| Prioridad | Alta |

Tabla 21: Requisito no funcional 1

|  |  |
| --- | --- |
| Nombre | Eficiencia |
| Descripción | * Toda funcionalidad del sistema y transacción de negocio debe responder al usuario en menos de 5 segundos. * Los datos modificados en la base de datos deben ser actualizados para todos los usuarios que acceden en menos de 2 segundos. |
| Prioridad | Alta |

Tabla 22: Requisito no funcional 2

|  |  |
| --- | --- |
| Nombre | Usabilidad |
| Descripción | * El sistema debe proporcionar mensajes de error que sean informativos y orientados a usuario final. * El sistema debe poseer interfaces gráficas intuitivas * La aplicación web debe poseer un diseño “Responsive” a fin de garantizar la adecuada visualización en ordenadores, tablets y dispositivos móviles. |
| Prioridad | Alta |

Tabla 23: Requisito no funcional 3

|  |  |
| --- | --- |
| Nombre | Mantenibilidad |
| Descripción | El sistema estará desarrollado de tal forma de que se adapte bien al cambio, es decir, ante cualquier cambio, mejora o inclusión de nuevas funcionalidades en el sistema no conlleve mucho trabajo ni tiempo. |
| Prioridad | Alta |

Tabla 24: Requisito no funcional 4

1. **Diseño y arquitectura**

El diseño es un parte fundamental de una aplicación web, puesto que el cliente es la parte del proyecto con la que va a interaccionar el usuario final. Por lo que tiene que ser intuitiva y fácil de usar y cumplir con sus expectativas.

A lo largo de éste apartado se mostrarán los diagramas realizados para el realizar el diseño y las funcionalidades que el usuario final podrá disponer. Además se muestra el diseño final y una explicación de la arquitectura del proyecto.

* 1. **Prototipo temprano**

Los *mockups* son una representación visual y estática de un diseño. Aunque permitan una representación del diseño mas cercana a la realidad que otros prototipos no suponen un diseño final, si no mas bien una aproximación.

Los *mockups* no hacen referencia a todas las vistas de la web, solo a las principales para tener una idea de la estructura principal de la aplicación web. Por esto, si los comparamos con el diseño final podemos comprobar que existen variaciones.

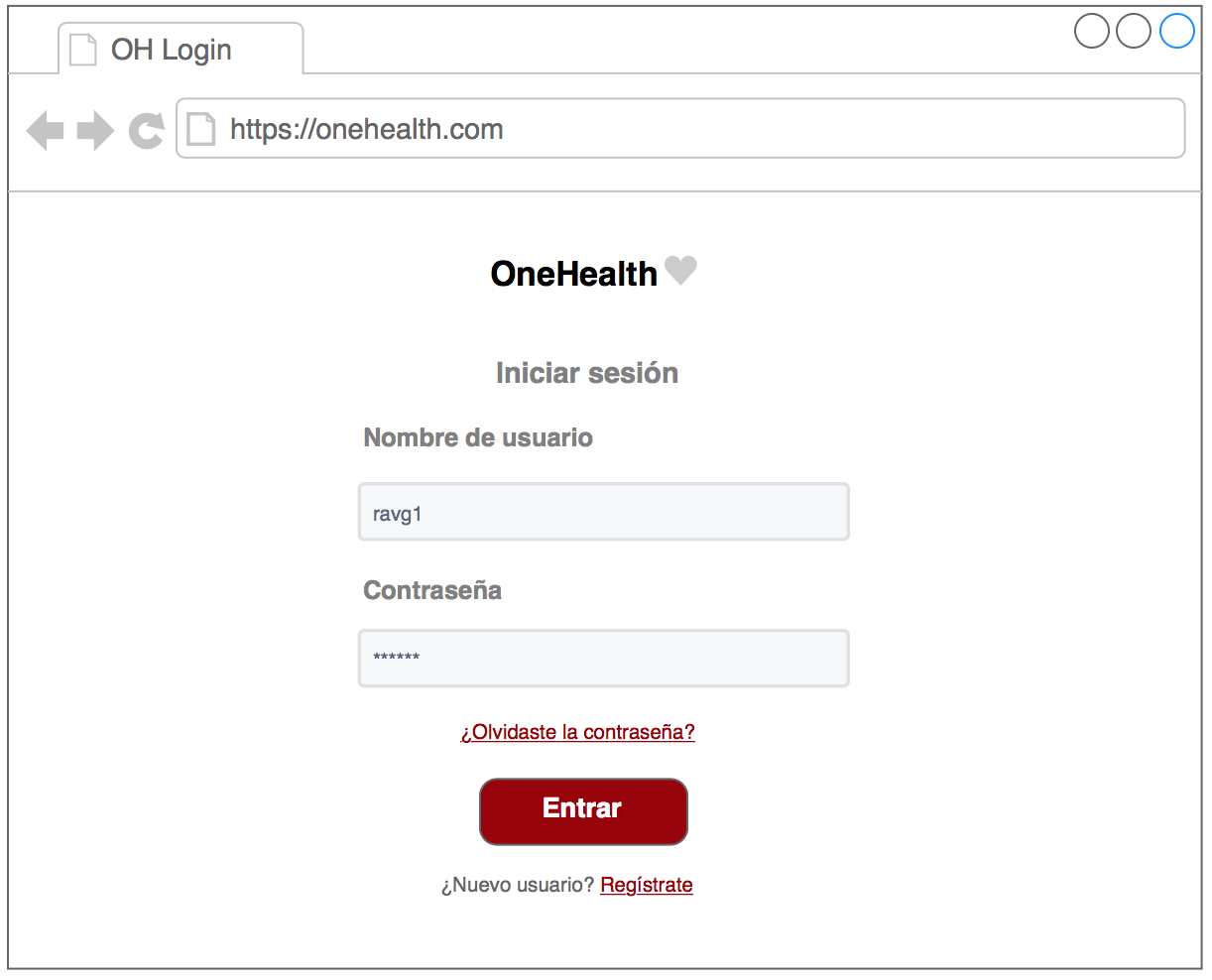
****

Ilustración 9: Mockup login

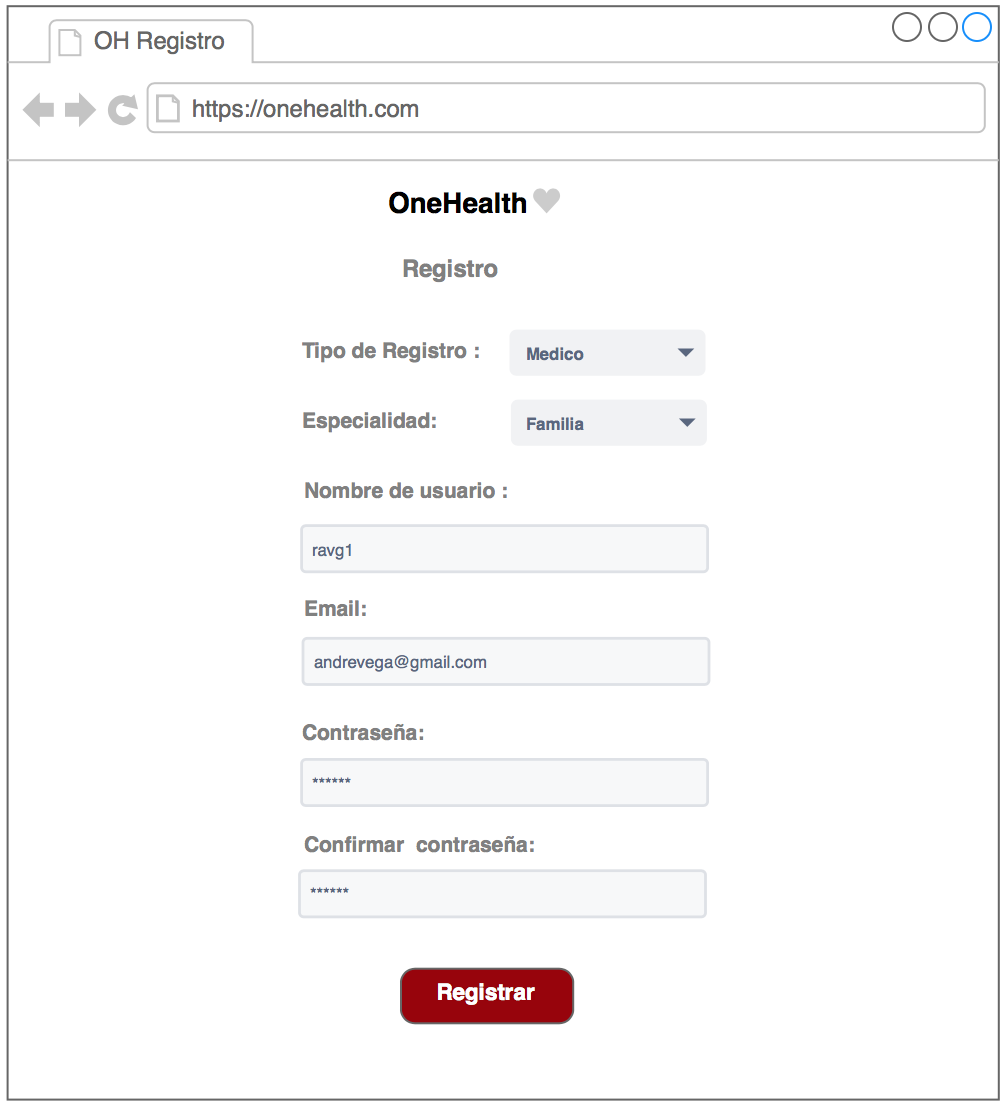


Ilustración 10: Mockup registro médico

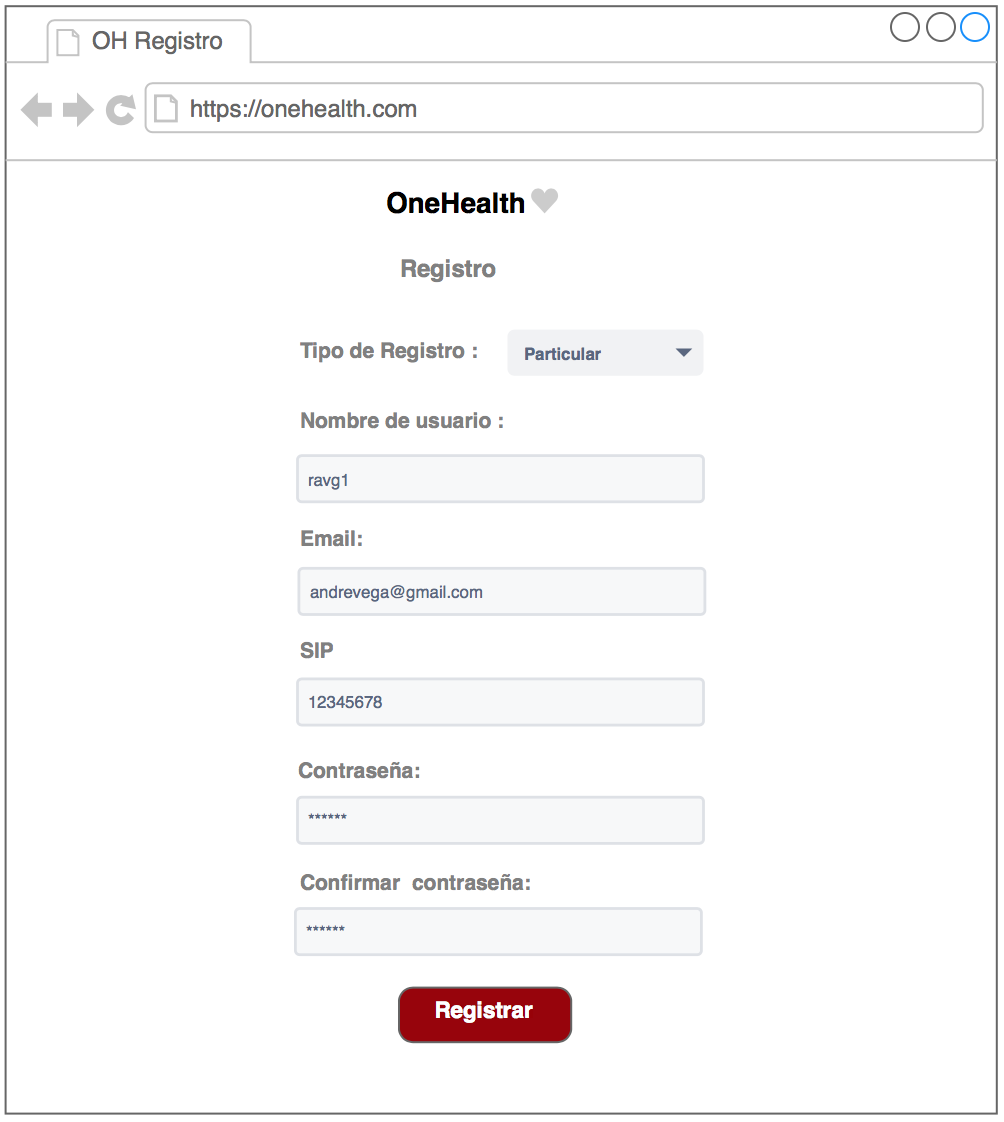


Ilustración 11: Mockup registro paciente

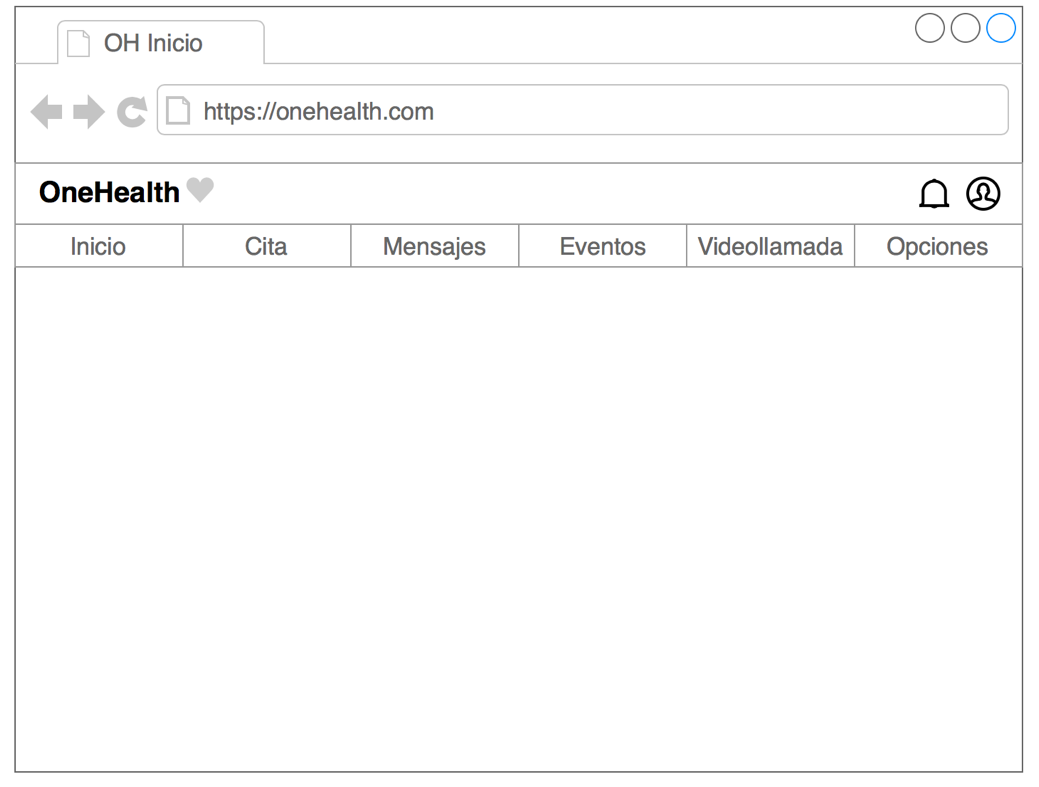


Ilustración 12: Mockup inicio

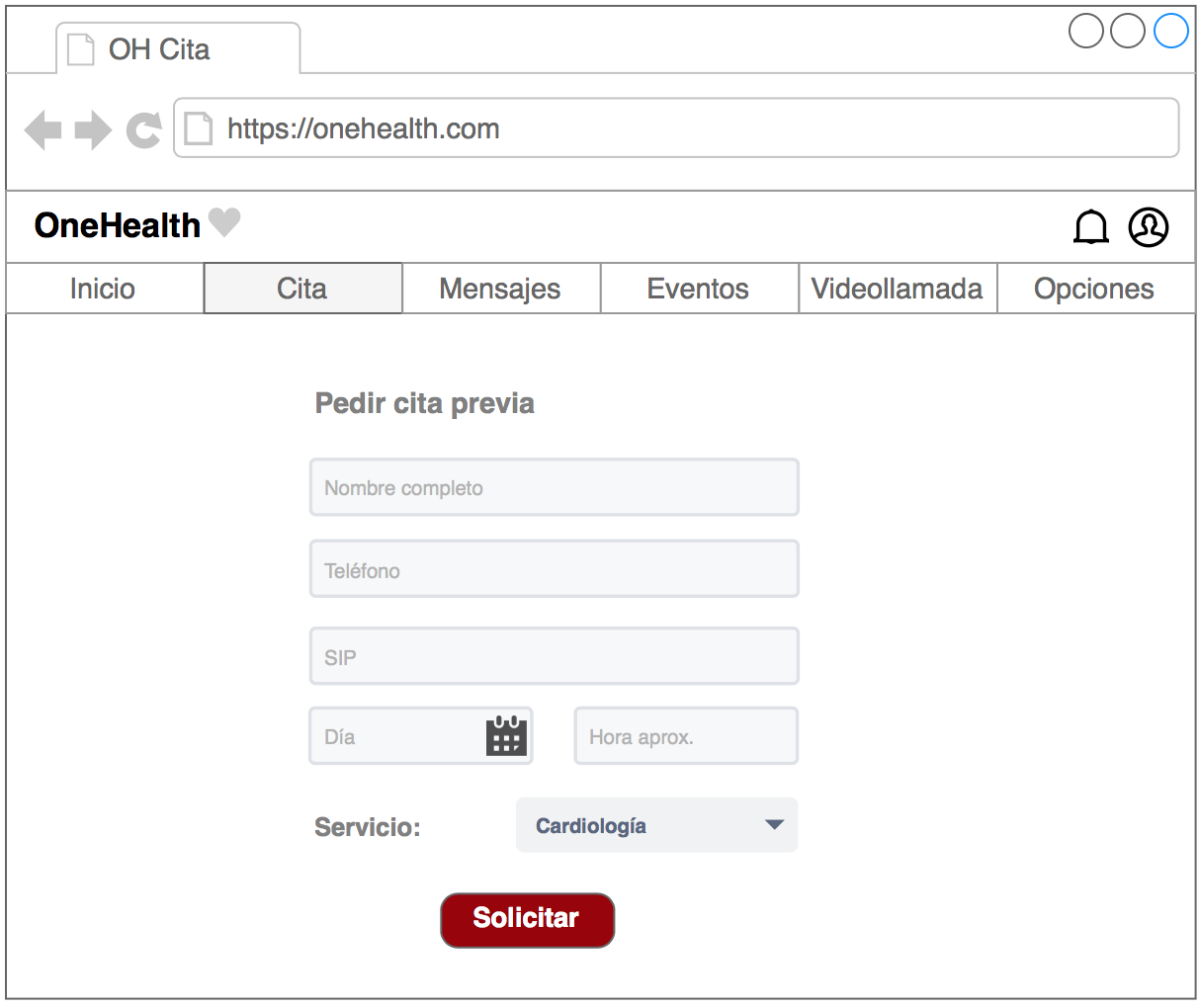


Ilustración 13: Mockup pedir cita

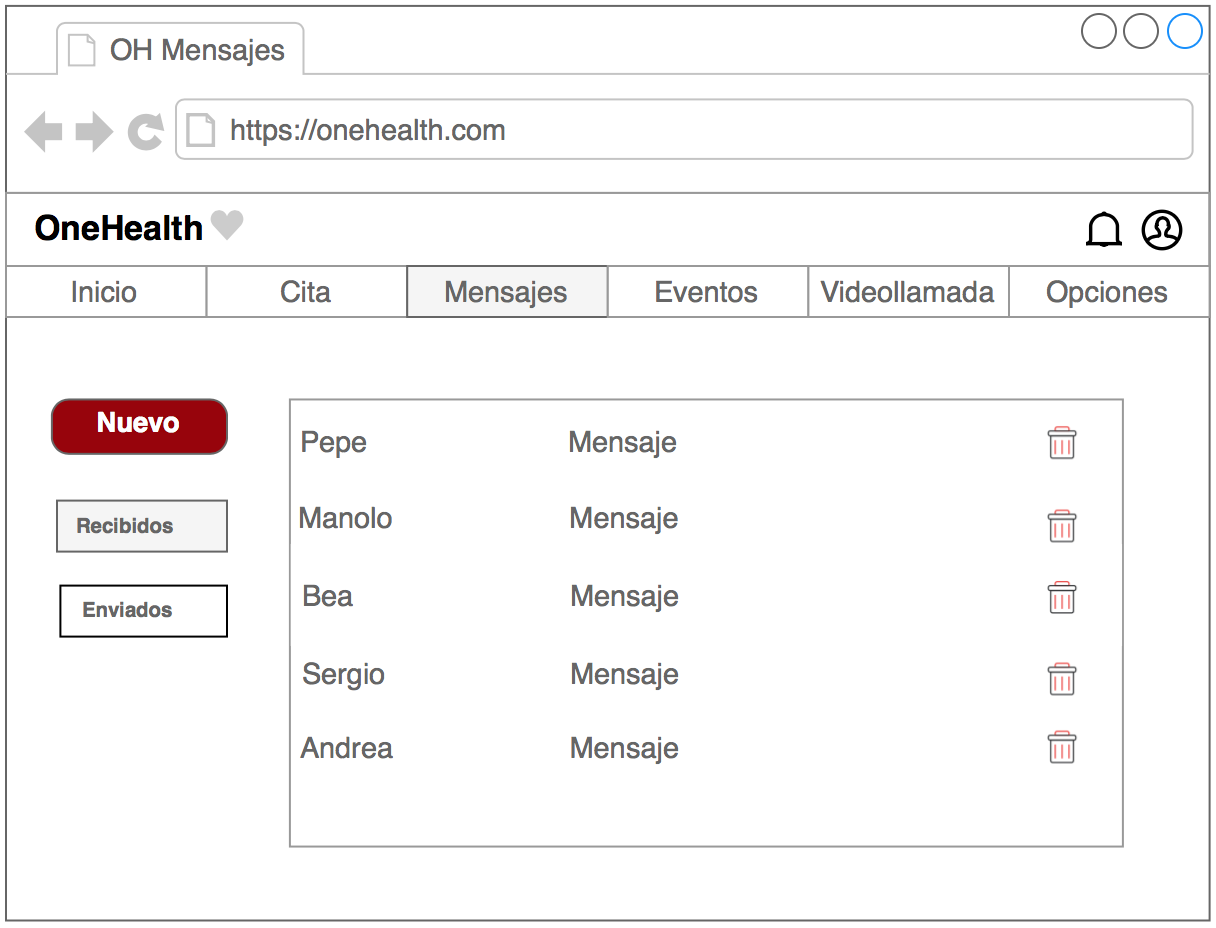


Ilustración 14: Mockup mensajes



Ilustración 15: Mockup historial

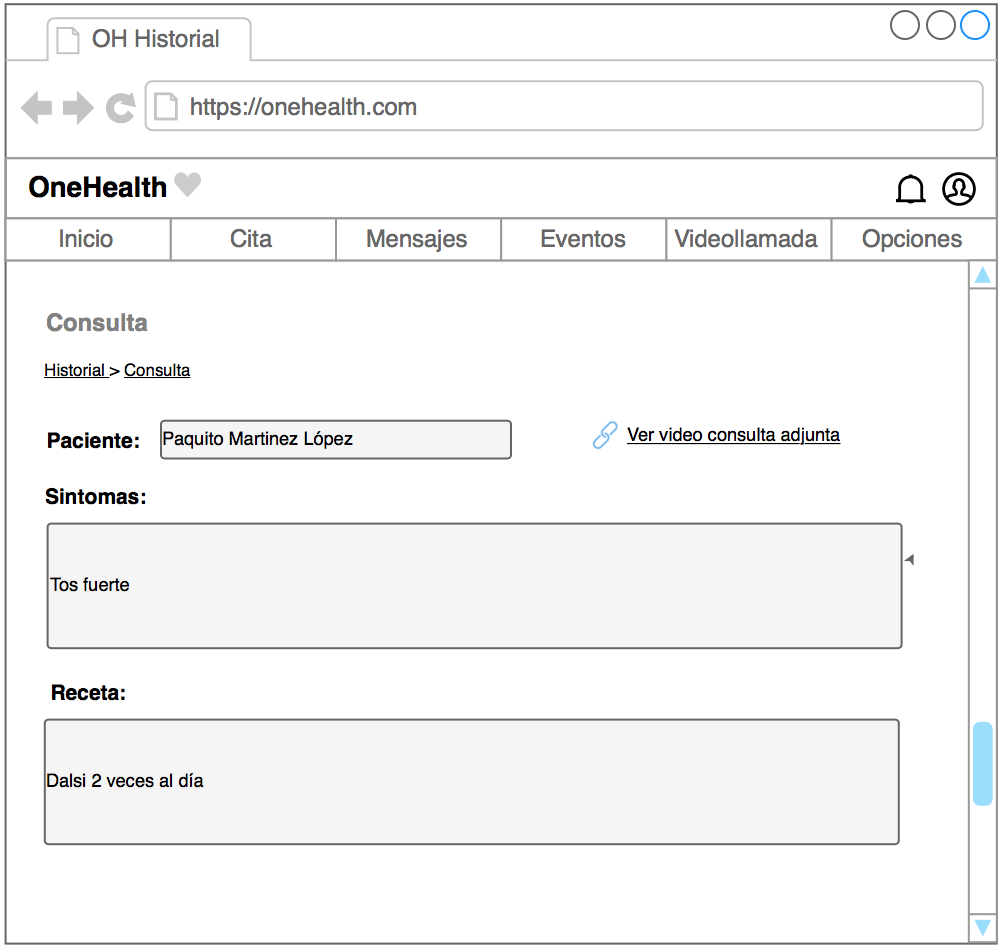


Ilustración 16: Mockup ver consulta

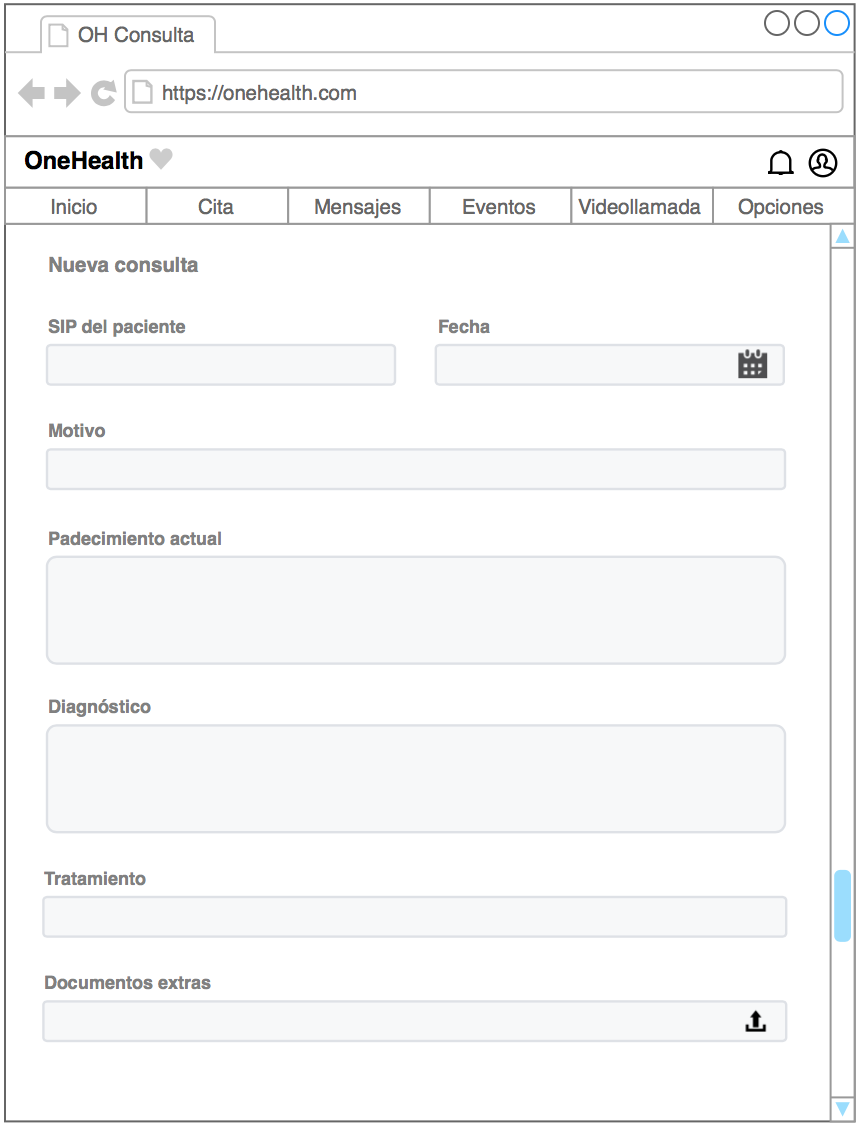


Ilustración 17: Mockup nueva consulta

* 1. **Diagrama de clases**

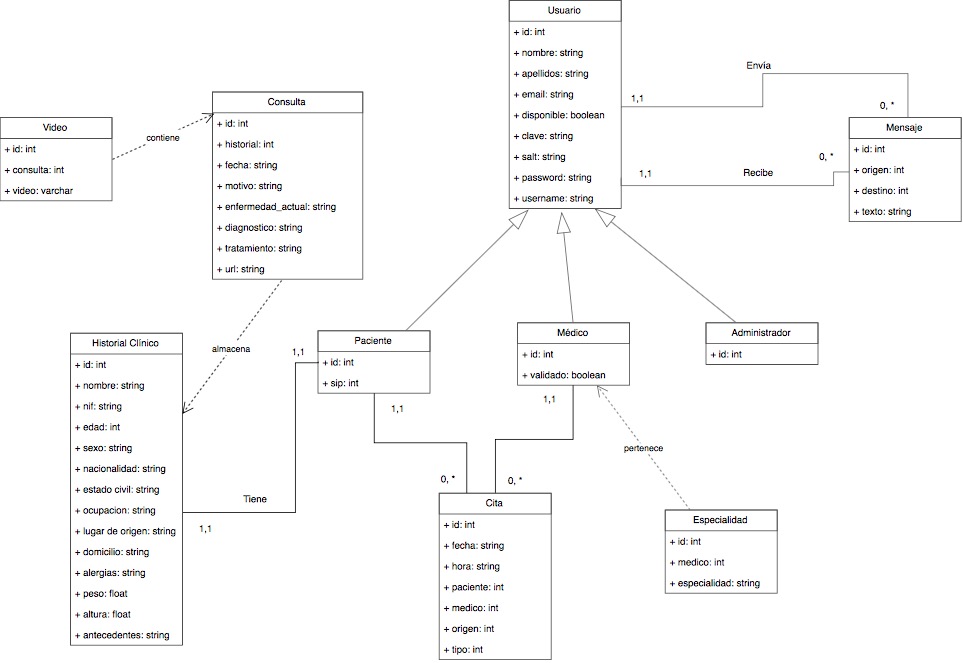
****

Ilustración 18: Diagrama de clases

* 1. **Diagrama entidad relación**

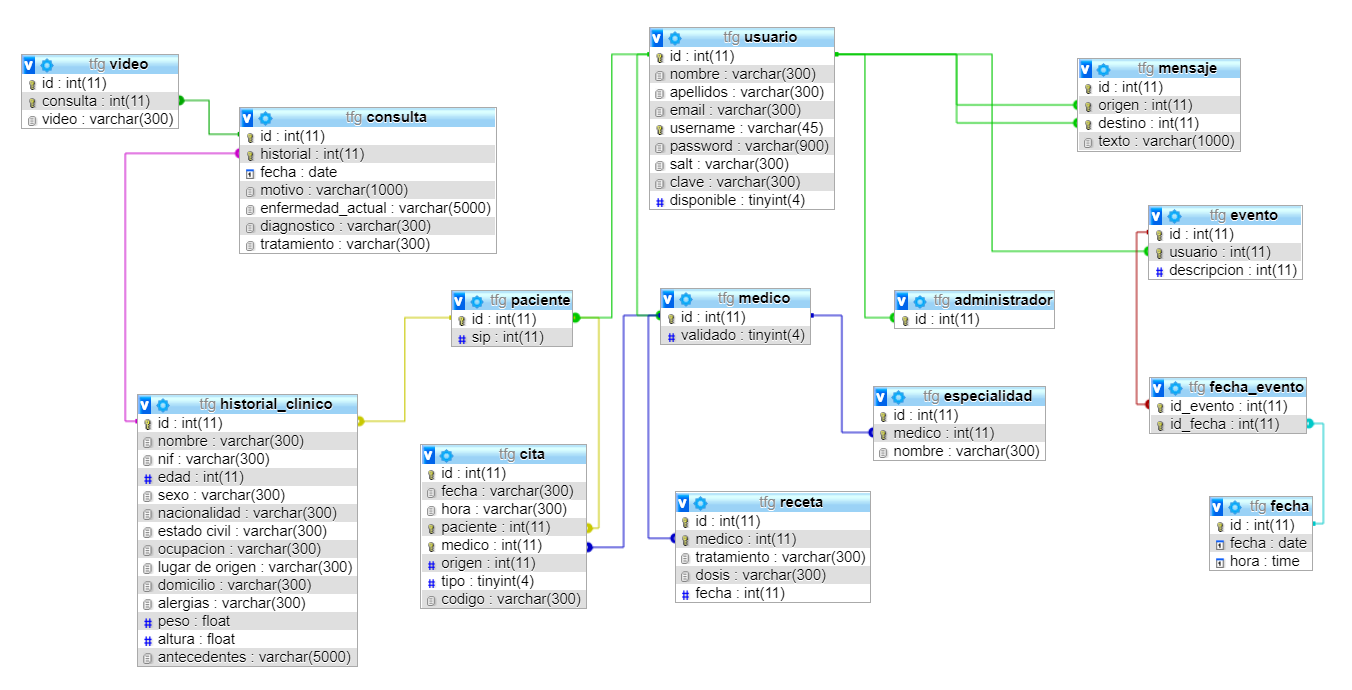


Ilustración 19: Diagrama entidad relación

* 1. **Diseño final**

Durante el desarrollo de la aplicación algunas vistas han sufrido cambios y reducciones haciendo mas simple alguna de ellas.

A continuación se mostrarán las vistas con el diseño final, incluyendo las que no estaban en los prototipos tempranos, además de una explicación con la secuencia del funcionamiento normal de la aplicación.

Las vistas de registro y *login* casi no han sufrido cambios con respecto a los prototipos, exceptuando en el registro, que se ha omitido el campo de confirmar contraseña. En ésta parte el usuario tiene que elegir que tipo de registro quiere, de tipo médico o de usuario normal. En el caso de que sea de tipo médico, cuando se registre no podrá hacer los servicios exclusivos de un médico hasta que el administrador no lo valide.

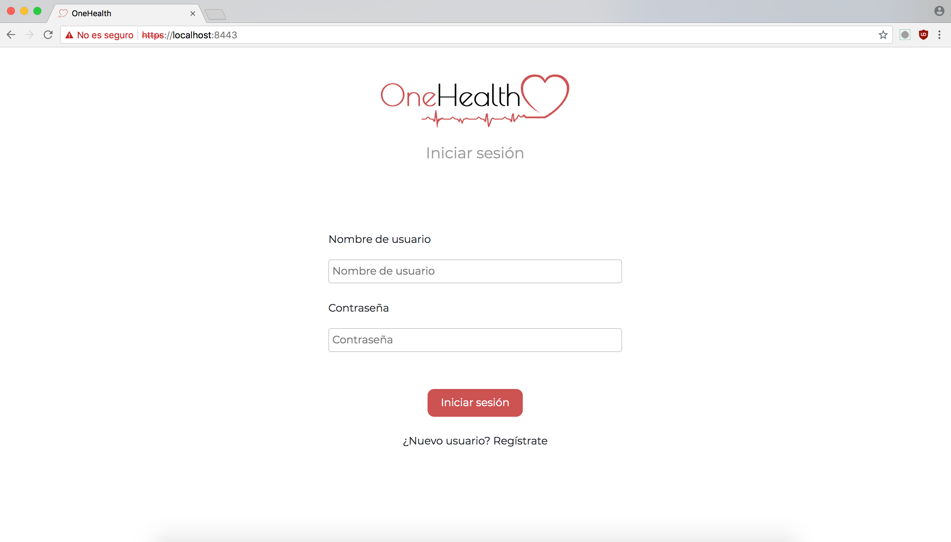


Ilustración 20: Login

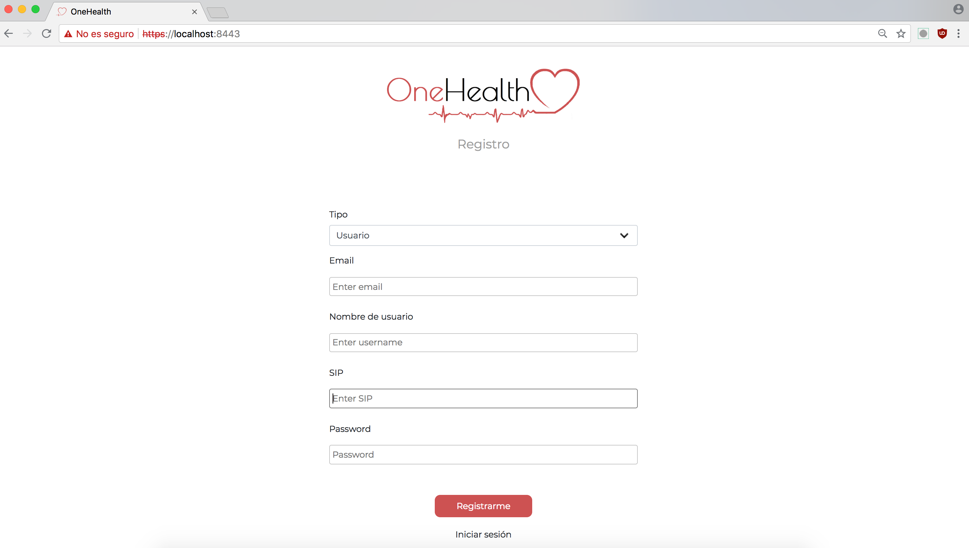


Ilustración 21: Registro

Una vez habiendo iniciado sesión, en la vista de inicio (*ilustración 22*) vemos que hay noticias sobre la salud, y que el menú ha cambiado con respecto a los prototipos. Se ha quitado la opción de eventos y opciones ya que por limitación de tiempo eventos no se ha podido implementar, posponiéndolo para futuras mejoras. Y en cuanto a la de opciones se optó por poner más opciones en el apartado “Perfil”.



Ilustración 22: Inicio

La primera opción que tenemos después de inicio es la citas, en ella tenemos dos alternativas, pedir una nueva cita o ver las citas. Las dos opciones pueden realizar tanto médico como paciente.

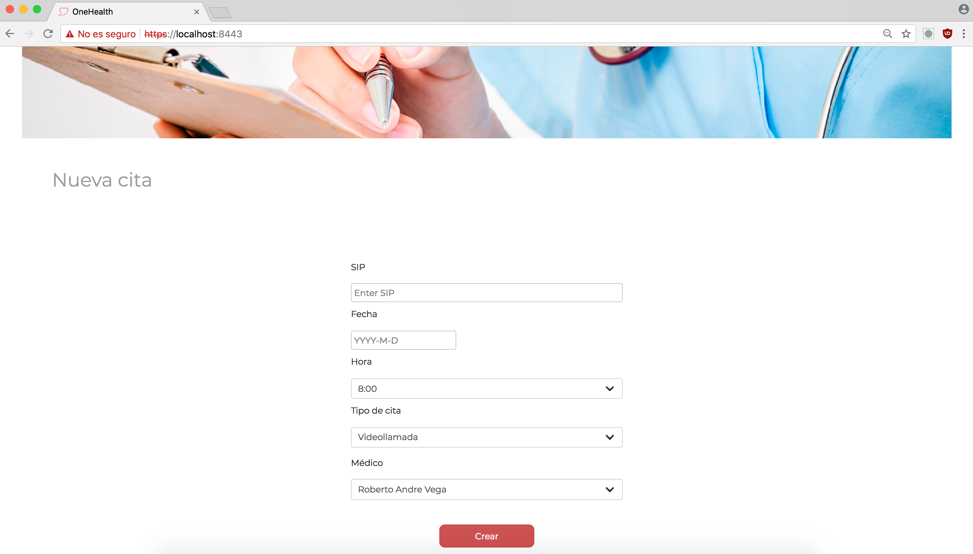


Ilustración 23: Nueva cita

En la opción de “nueva cita” (*Ilustración 23*) el usuario tiene que introducir la SIP del paciente, la fecha, seleccionar una hora, elegir el tipo de cita, ésta puede ser videollamada o presencial si es necesario y por ultimo el nombre del médico con el que concretar la cita. Una vez rellenados los campos correctamente se le proporciona un código de la cita al usuario, que luego llegada la fecha de la cita lo tendrá que introducir para comprobar que la cita es válida y esta dentro del rango de la fecha y hora establecido cuando se creó la cita.

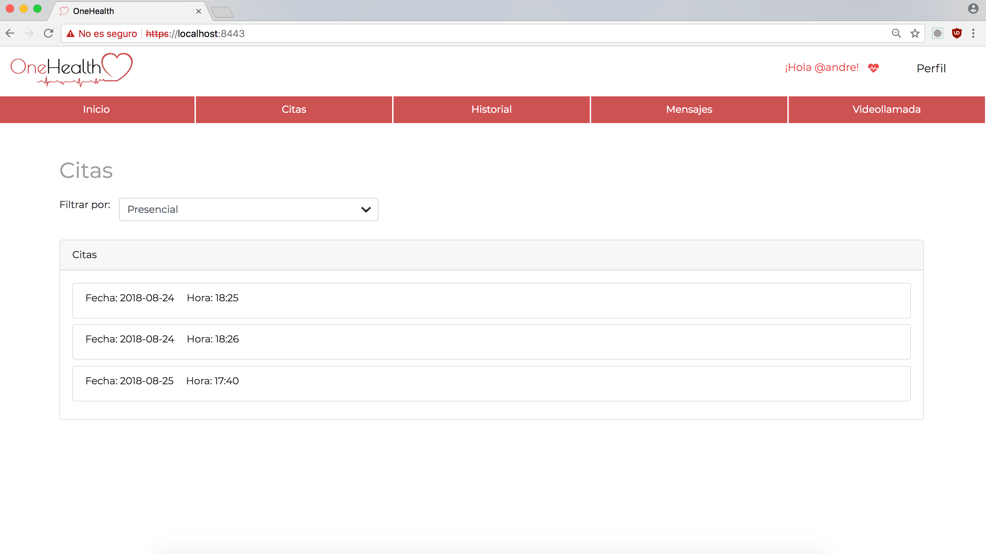


Ilustración 24: Ver citas

En la opción de ver citas, el usuario puede ver todas sus citas, parecido a un historial de citas, y filtrarlas según sean de tipo de videollamada o presencial.

La siguiente opción en el menú es la historial, aquí las alternativas varían según el tipo de usuario que sea, si es médico tiene la opción de buscar el historial de un paciente mediante su SIP y crear una nueva consulta asociada al historial de un paciente.

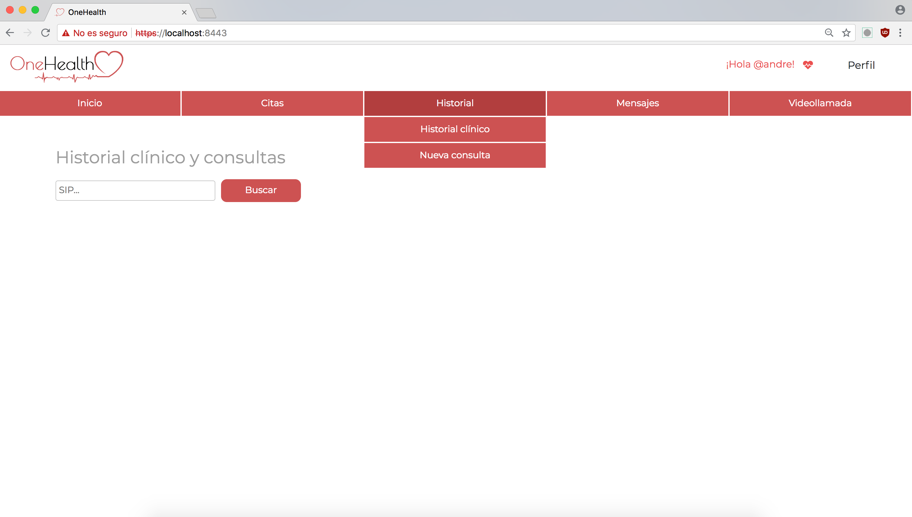


Ilustración 25: Historial según la vista del médico

Cuando el médico busca un historial clínico, le aparecen los datos del paciente y las consultas asociadas a ese historial.

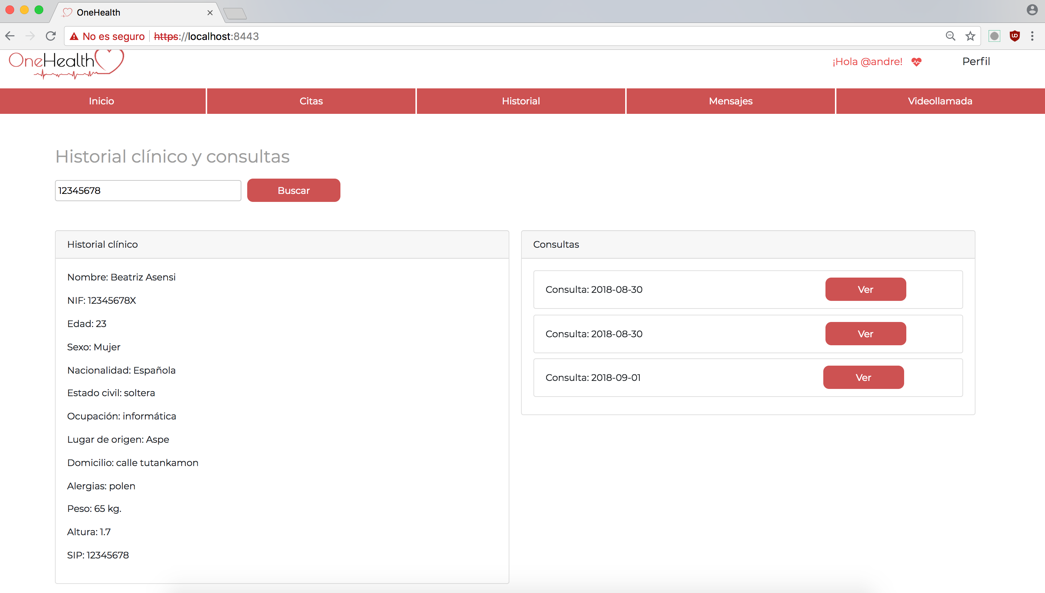


Ilustración 26: Historial y consultas

El médico tiene la posibilidad de acceder a todas las consultas, dentro de cada consulta le aparecerán la información con respecto a esa consulta, incluso un video de la videollamada en caso de que fuera de ese tipo.

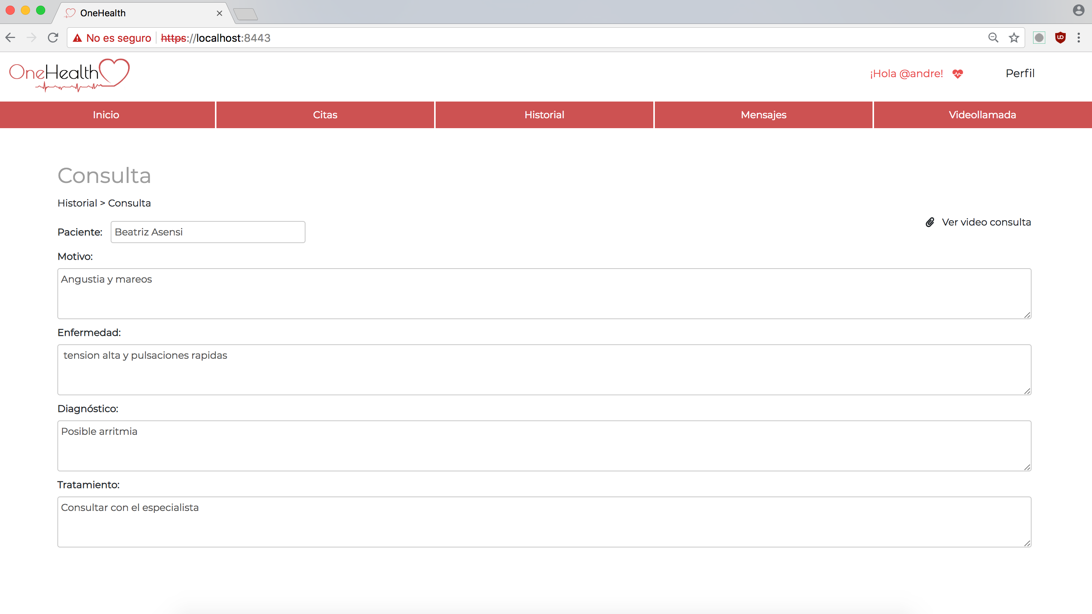


Ilustración 27: Ver consulta

Para crear una nueva consulta el médico tiene que rellenar los campos, siendo obligatorio el de la tarjeta SIP (*Ilustración 28)* .

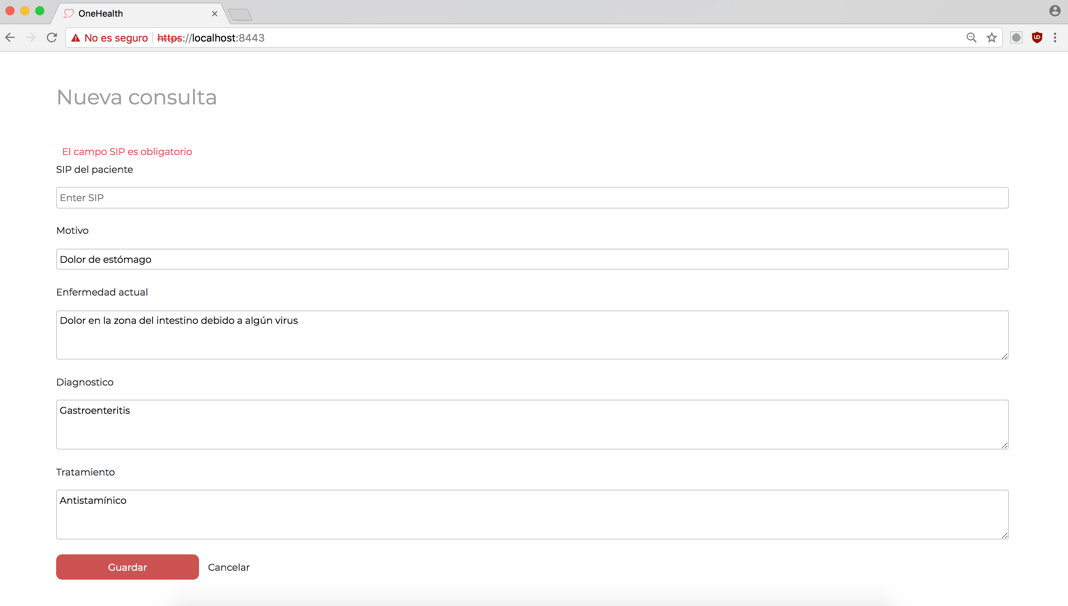


Ilustración 28: Nueva consulta

Si es un paciente, al hacer clic en la opción de historial, simplemente le aparecerán los datos personales de su historial exceptuando las consultas.

(imagen)

La siguiente opción es la de mensajes, dicha opción esta disponible tanto para pacientes como para médicos. Paciente y médico podrán comunicarse a través de mensajes. Cada uno podrá enviar un nuevo mensaje, poniendo el nombre de usuario del destinatario, un asunto y el mensaje, todos campos obligatorios. Y también podrán ver sus mensajes recibidos y enviados, pudiendo borrar los enviados.

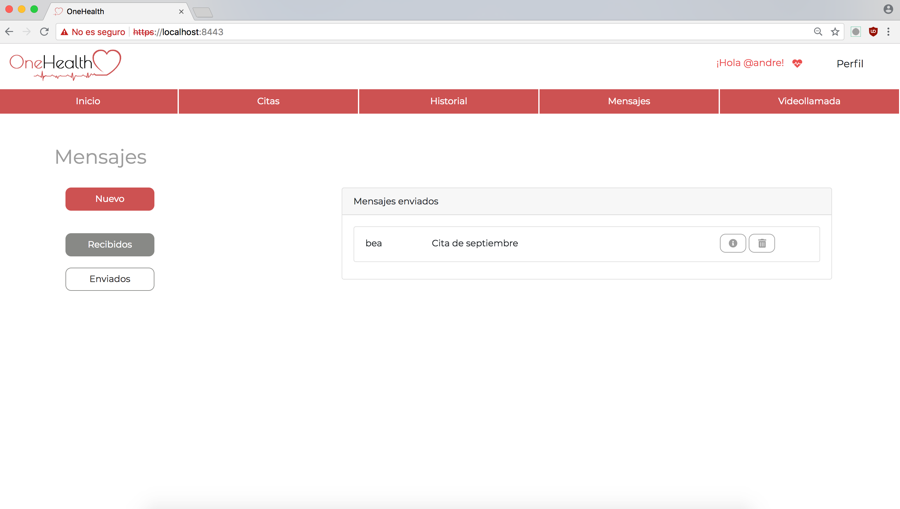


Ilustración 29: Mensajes (enviados)

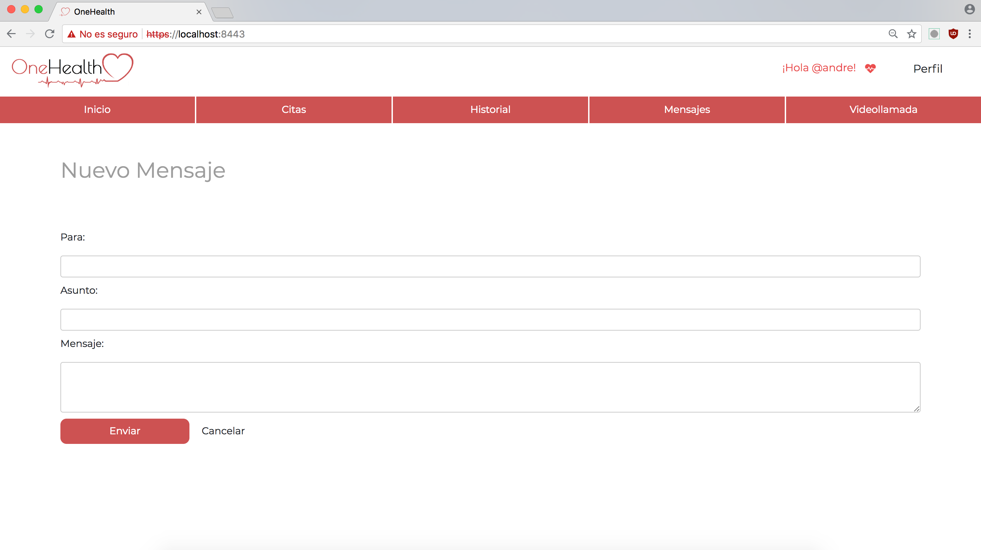


Ilustración 30: Nuevo mensaje

Y por ultimo la opción de videollamada, que varía según el usuario sea médico o paciente. Si el usuario es de tipo paciente, se le pedirá el código de la cita que se le proporcionó al crear la cita, si es válido podrá empezar la videollamada. En el caso de que sea médico le aparecen un listado de pacientes conectados con la opción de poder llamarlos y un buscador para buscar un paciente mediante la tarjeta SIP.

* 1. **Arquitectura**

La arquitectura de la cual se compone el proyecto, es una arquitectura cliente-servidor, la cual es un modelo de comunicación que vincula varios dispositivos a través de una red. El cliente realiza peticiones al servidor y éste se encarga de responder a dichas peticiones, lo que posibilita que las tareas se distribuyan entre los servidores y los clientes. Éste tipo de modelo permite que el servidor pueda ejecutarse en mas de un equipo lo que facilita repartir la capacidad de procesamiento.

La arquitectura del proyecto es de dos capas, ya que el servidor accede directamente a los datos y se los devuelve al cliente que hizo la petición.

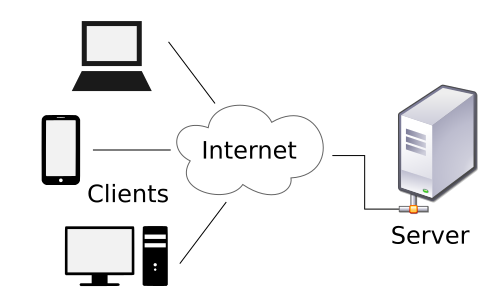


Ilustración 31: Arquitectura cliente-servidor (dos capas)

1. **Seguridad**

En la actualidad la mayoría de la información se gestiona a través de internet, y muchos de esos datos suelen ser importantes para el usuario, como por ejemplo contraseñas, tarjetas de crédito, datos sanitarios.. Por eso, la seguridad en el sistema es uno de los principales propósitos a conseguir para el proyecto. Al trabajar con datos sensibles de usuarios es imprescindible mantener la integridad de dichos datos y dar una imagen de un sistema seguro, ya que, una mala gestión de la seguridad puede tener tanto impacto económico como afectar a la reputación y la confianza de socios y clientes.

En éste apartado se detallarán los aspectos de seguridad, tanto en el servidor como en el cliente, que se han implementado en el proyecto.

* 1. **Servidor**

Del lado del servidor se han implementado las siguientes características relacionadas con la seguridad:

* **Conexión:** para asegurar la transferencia segura de datos se ha usado el protocolo HTTPS, el cual es un protocolo de aplicación basado en HTTP pero destinado a la transferencia segura de datos.

HTTP es el protocolo que se utiliza para la comunicación entre cliente y servidor pero tiene el problema que la transferencia de datos se hace en plano, es decir, sin cifrar. Por eso en el proyecto se ha optado por usar HTTPS, el cual se podría decir que es la versión segura de HTTP, ya que esta basado en SSL/TLS.

SSL/TLS (Secure Sockets Layer/Transmission Layer Security) son dos protocolos para enviar paquetes cifrados a través de internet. Ambos utilizan lo que se conoce como un sistema de infraestructura de clave pública (PKI) asimétrico. Un sistema asimétrico utiliza dos claves para encriptar comunicaciones, una clave pública y una clave privada. Cualquier cosa cifrada con la clave pública solo se puede descifrar con la clave privada y viceversa.

Con esto HTTPS crea un canal cifrado para la transferencia de información sensible, lo que significa que esos datos del usuario están cifrados y no pueden ser interceptados. Otra ventaja es que los visitantes pueden verificar el dueño legitimo del dominio y así poder confiar más en el sistema.

* **Certificados:** como se ha comentado anteriormente los certificados SSL sirven para establecer un canal seguro entre emisor y receptor, el problema es que algunos sitios no utilizan los certificados realmente seguros a día de hoy.

Los certificados SSL son firmados utilizando un hash de una única dirección, para garantizar la integridad del mismo (y garantizar que el certificado no se ha modificado). Siguen existiendo en internet sitios que utilizan SHA-1. Y es que SHA-1 es inseguro, debido a vulnerabilidades matemáticas, ya que existen colisiones que permiten *crackear* el protocolo 2000 veces más rápido de lo esperado (*Schneier: Cryptanalysis of SHA-1*).

La solución es usar un algoritmo de hash seguro, como SHA-2, en los certificados. En nuestro caso, en el proyecto se ha optado por usar SHA512 con el algoritmo de encriptación RSA.

RSA es un sistema criptográfico de clave publica, se considera uno de los algoritmos mas seguros ya que nadie ha conseguido demostrar o rebatir su seguridad. Las claves publicas y privadas se calculan a partir de un número que se obtiene como producto de dos números primos grandes, por lo que un supuesto atacante se enfrenta a un problema de factorización de un coste computacional muy elevado.

* **Sistemas de autenticación**: en cuanto a la autentificación de los usuarios, en el proceso de *login* hay un **doble factor de autenticación**. Primero el usuario tiene que introducir su nombre de usuario y contraseña, y si todo está bien, se le envía un correo electrónico con un código generado aleatoriamente en el servidor, el usuario tiene que introducirlo, se comprueba su validez y ya puede iniciar sesión en el sistema.

Para la autenticación de los médicos, el médico en cuestión se tiene que registrar con normalidad eligiendo la opción correcta en el registro. Una vez registrado en el sistema para poder tener acceso a los servicios tiene que ser validado antes por un administrador del sistema, que se encargará de comprobar que en realidad es un médico cuando este aporte la documentación necesaria para corroborarlo.

Cuando el usuario ya ha iniciado sesión en el sistema, se utiliza la autenticación basada en ***Token*.** Ésta autenticación consiste en que cada petición HTTP que haga el usuario va acompañada de un *Token* en la cabecera. Este *Token* no es más que una firma cifrada que permite a nuestro API identificar al usuario, pero no se almacena en el servidor, si no en el lado del cliente y el API es el que se encarga de descifrar ese *Token* y redirigir el flujo de la aplicación en un sentido u otro. Esto nos añade más seguridad, al no utilizar cookies para almacenar la información del usuario, podemos evitar ataques CSRF (Cross-Site Request Forgery) que manipulen la sesión que se envía al *backend*.

En el proyecto se ha optado por usar ***JSON Web Token (JWT)****.* Este *Token* esta compuesto por tres partes: la cabecera, que contiene el tipo (JWT) y la codificación utilizada (SHA256), el *payload* que es donde van los atributos del *Token,* que en nuestro caso son los siguientes:

var payload = {

sub: user.id,

iat: moment().unix(), //cuando se ha creado el token

exp: moment().add(15, "days").unix() //cuando expira

}

Y la ultima parte es la de *signature,* la cualestá formada por los anteriores componentes (*Header y Payload*) cifrados en Base64 con una clave secreta (almacenada en nuestro API). Así sirve de Hash para comprobar que todo está bien*.*

* **Métodos de cifrado:** en el proyecto se cifran todos los datos de los usuarios antes de ser almacenados en la base de datos con una clave única para cada usuario, generada aleatoriamente cuando el usuario se dio de alta en el sistema. El algoritmo que se ha utilizado es AES256 combinado con CBC. AES es un estándar de cifrado avanzado (*Advanced Encryption Standard*), que hoy en día es considerado uno de los cifrados más seguros y utilizados. El algoritmo se basa en varias sustituciones, permutaciones y transformaciones lineales, ejecutadas en bloques de datos de 16 bytes. Estas operaciones se repiten varias veces, llamadas “rondas”.

El modo de operación es CBC, el cual opera de forma que a cada bloque de texto se le aplica una operación XOR con el bloque previo ya cifrado. De este modo, cada bloque cifrado depende de todos los bloques de texto claros usados hasta ese punto. Además, se utiliza un vector de inicialización para que cada mensaje sea único.

* 1. **Cliente**

En el lado del cliente, gracias a la herramienta de *Webpack*, se usa también el protocolo HTTPS con certificados SSL SHA256, con el algoritmo de encriptación RSA, tal y como se hace en el servidor.

1. **Implementación**

Como ya se ha comentado anteriormente el proyecto se divide en dos grandes partes, por un lado el API y por otro el cliente, por lo que para hablar de la implementación del sistema es necesario tener las dos partes en cuenta.

* 1. **API**

El API como ya se ha dicho en otros apartados se ha desarrollado con *NodeJs* utilizando peticiones HTTPS con los métodos de petición para indicar la acción que se desea realizar para un recurso determinado.

Para la implementación es necesario saber el funcionamiento del sistema y el modelo de datos (*Ilustración 18*), para saber todos los elementos del proyecto y como interacción entre ellos.

Primero hay que destacar que todas las funciones del sistema giran en torno al usuario, ya que tiene relación con todas las acciones disponibles. Hay tres tipos de roles de usuario:

* **Paciente**: es el usuario estándar del sistema, tiene acceso a la mayoría de funciones, exceptuando la de ver las consultas y los videos asociados a la consultas y la de crear una nueva consulta.
* **Médico**: es el usuario que tiene acceso a todas las funciones disponibles del sistema, con la diferencia del paciente de que tiene que ser validado por un administrador antes de poder tener acceso a dichas funciones.
* **Administrador**: el administrador del sistema se encarga de validar los nuevos médico registrados y añadir los historiales clínicos de los nuevos pacientes.

Para controlar que cada usuario tenga acceso a sus respectivas tareas, se ha hecho uso de un *middleware* que controla las diferentes rutas de las funciones. Pero también existe el problema de que un usuario intente acceder a una ruta usando el nombre de usuario de otro usuario con otro rol diferente al suyo, para esto se ha implementado también un *middleware* que comprueba que ese usuario esta conectado al sistema*.* Las únicas rutas que no están bajo este control son las de registro, *login*, confirmación de *login* y listar especialidades (se usa en el registro del médico para poder listar todas las especialidades disponibles).

* + 1. **Login y registro**

Como seguridad en el registro se obliga al usuario a poner un nombre de usuario que no exista un correo electrónico valido y su tarjeta SIP, en el caso de que sea un paciente, además de una contraseña. Esta contraseña en el servidor no se guarda en claro en la base de datos, si no que se le aplica una codificación con SHA512. Si todo ha ido bien, se guarda el usuario y se genera una clave aleatoria que luego se usará para el cifrado de datos.

En la parte del *login* cuando el usuario introduce correctamente el nombre de usuario y contraseña, se genera un código aleatorio de cuatro números que se le envía al correo electrónico que puso en el registro. Una vez que el usuario haya introducido dicho código y se haya validado, se genera un *token* único para cada usuario, ya que se incluye el id de cada usuario, y con un tiempo de expiración de quince días. Cada vez que el cliente haga una petición debe incluir este *token* en la cabecera de *Authorization* y en el servidor se comprueba su validez. Cada vez que el usuario inicia sesión se actualiza el *token* y cuando se cierra la sesión se elimina.

var payload=jwt.decode(token, config.TOKEN\_SECRET)

if(payload.exp <= moment().unix()) {

return resp.status(401).send({message: "El token ha expirado"})

}

* + 1. **Historial**

En el apartado del historial clínico es diferente según el usuario, en el caso de que sea un paciente, se busca el historial según la id del usuario conectado, se desencriptan los datos con la clave de dicho usuario, y se devuelven solo los datos personales sin las consultas asociadas.

Si el usuario es de tipo médico se hace la petición a otra ruta donde se comprueba que efectivamente es un médico y no de otro tipo. En este caso el historial se busca mediante la tarjeta SIP del paciente y se devuelven todos los datos del historial así como todas las consultas registradas a dicho historial.

Por ultimo en el caso del administrador una de sus funciones es la de añadir un nuevo historial, dentro de esta opción aparece un formulario con los datos necesarios del historial clínico, para identificar al usuario el administrador debe introducir, como campo obligatorio, el *id* con el que se ha almacenado ese usuario en la base de datos. Una vez introducidos los datos se hace la petición y en el servidor se cifran los datos con la clave del usuario introducido.

* + 1. **Consultas**

Las consultas son exclusivas del médico, cuando se busca un historial el servidor devuelve automáticamente las consultas asociadas, en el caso de tenerlas, para que el medico ,si quiere, pueda verlas y así no hacer peticiones de mas al servidor.

La otra opción disponible es la crear una nueva consulta, en ella el médico tiene que rellenar unos campos con los datos de la consulta, siendo únicamente obligatorio el de la tarjeta SIP para poder identificar al usuario y con ello su historial, para así cifrar los datos y poder guardar la consulta correctamente.

* + 1. **Mensajes**

El servicio de mensajes es igual tanto para médico como para paciente, pueden ver los mensajes recibidos y enviados, enviar un nuevo mensaje y borrar los mensajes. Los métodos de ver recibidos y ver enviados son muy parecidos, el de recibidos busca los mensajes donde el destinatario sea el usuario conectado, y descifra los datos con la clave del remitente del mensaje, es decir, el que lo envió. Y el de ver enviados simplemente busca los mensajes donde el remitente sea el usuario conectado, descifrando los mensajes con su propia clave, ya que él fue el que lo creó.

Para enviar un nuevo mensaje el usuario tiene que poner el nombre de usuario del destinatario de forma correcta, de lo contrario el método devolverá un error diciendo que no se ha encontrado al destinatario. Además de eso, tiene que poner el asunto del mensaje y el mensaje en si, ambos campos obligatorios. Los datos en el servidor se cifran con la clave del usuario que ha enviado el mensaje.

El usuario también tiene la opción de borrar un mensaje, pero esta función esta limitada a los mensajes enviados, ya que se comprueba que el usuario que quiere borrar el mensaje sea el que originalmente lo envió.

* + 1. **Citas**

El servicio de citas incluye las funciones de crear una nueva cita, ver las citas y borrar una cita.

El usuario puede ver todas las citas, o bien filtrarlas por el tipo de cita que sea, es decir, videollamada o presencial. Los métodos para listar la citas están organizados para listar las citas de un paciente y sus dos tipos, y para listar las citas de un medico y sus dos tipos. Es decir, que según sea medico o paciente se manda la petición a unos métodos u otros. Cada método descifra los datos de las citas con la clave del usuario que haya creado la cita y devuelve una información general sobre las citas. Para poder ver los detalles de cada cita se llama a otro método especifico al que se le pasa el *id* de una cita y que devuelve los detalles de esa cita.

Para crear una cita el usuario tiene que rellenar todos los campos que le aparecen en el formulario, ya que en este caso todos son obligatorios. La cita registra a un paciente identificado por la tarjeta SIP, un médico al que se tiene que elgir entre la lista de médicos disponibles que se ofrece y una fecha y hora concretos. Los datos en servidor se cifran con la clave del usuario que ha creado la cita. Una vez creada con éxito el servidor devuelve como respuesta un código que el usuario debe guardarse para que llegado el día y la hora, en el apartado de videollamada pueda entrar a la cita.

Si el usuario quiere cancelar una cita, tiene que irse al apartado de videollamada e introducir el código de la cita y en él le aparece la opción de cancelar la cita, se hace la petición pasando el *id* de la cita y se borra de la base de datos.

* 1. **Cliente**

El cliente se ha desarrollado con *ReactJS,* que como se ha comentado anteriormente es una librería para crear interfaces basadas en componentes.

En la *ilustración 31* podemos observar la estructura del proyecto de la parte del cliente. El archivo “index.html” es el único archivo *html* y es donde se colocan los componentes, en nuestro caso esta el componente ”main.js”.

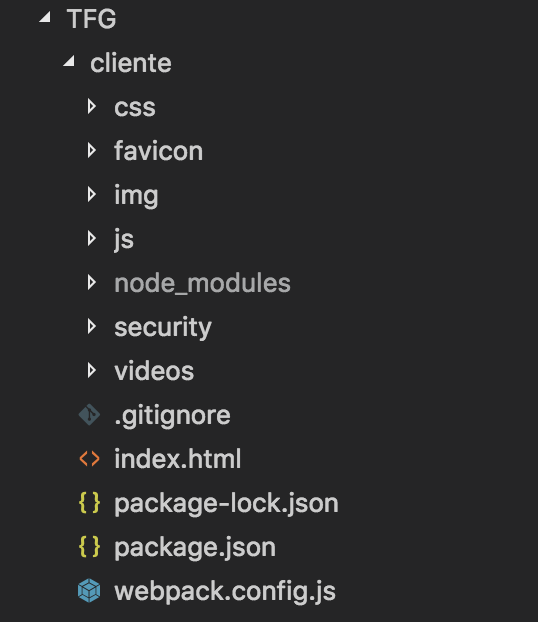


Ilustración 32: Estructura del proyecto (cliente)

Los componentes están en la carpeta “js”. En el componente *main* es donde están las librerías necesarias y donde se renderiza el primer componente. Este es el componente “index” el cual controla los estados de que indican si el usuario ha iniciado sesión, si esta *logueado* o si tiene que introducir el código de confirmación. Según sea el caso se renderiza el componente de inicio, login o confirmación de *login*.

var token = localStorage.getItem('token')

if (!this.state.logged && token==null && !this.state.confirmado){

return <Login handleConfirmacion={this.confirmacion}/>

}else if(!this.state.logged && token==null && this.state.confirmado){

return <Confirmar handleLoginOK={this.loginOK}/>

}else{

return <Inicio handleLogout={this.logoutOK}/>

}

* 1. **Registro y login**

En el componente *login* se hace la petición HTTPS y si es correcta, se llama a una función del componente padre para que cambien el estado y renderice el componente inicio. También hay una opción mediante un enlace para renderizar el componente registro.

En el registro al tener dos opciones posibles de formulario, para el médico o para el paciente, hay un estado que va cambiando según el usuario selecciona un tipo u otro. En el caso del formulario del medico hay que listar las especialidades disponibles para que le aparezcan en el selector al usuario, para ello llamamos a la función que hace la petición desde el método “componentDidMount()” que es llamado automáticamente después del *render.*

componentDidMount(){

this.obtenerEspecialidades()

}

Si el registro ha sido correcto, se llama a una función del componente padre (*login*) y este a su vez llama a otra función del padre (index) para que renderice el componente inicio.

* 1. **Inicio**

El componente inicio es el que controla todas las opciones del menú de la pagina de inicio (*ilustración 22*) mediante estados. Además, también controla lo que devolver según el tipo de usuario que haya iniciado sesión. En el caso de que sea el administrador, se renderiza el componente administrador donde la vista cambia por completo, mostrándole solo sus funciones disponibles. En el caso del medico en el historial le aparece un desplegable con las opciones de historial y nueva consulta, las cuales al paciente no le aparecen.

* 1. **Historial**

Dentro del componente historial se controla la información que se devuelve según el tipo de usuario que sea. Si es un médico, se muestra una vista para que pueda buscar el historial de un paciente, si la petición se ha hecho con éxito, se cambia el estado y se muestra el historial resultado de la busque y sus consultas, en el caso de tenerlas. En el caso de que sea de tipo paciente, se necesita el historial antes de que se llame al *render,* ya que cuando el paciente haga clic en la opción del historial ya le tienen que aparecer sus datos, por ello, se inicializa una variable en el constructor al resultado de la función que hace la petición para obtener el historial según el *id* del usuario conectado.

* 1. **Consultas**

Los componentes que intervienen en las consultas son el de listar las consultas y ver una consulta. El de listar consultas se llama desde el de historial y se encarga de recorrer el *array* de consultas que se le pasa desde historial y para cada consulta llamar al componente de ver consulta, al que se le pasa los datos de una sola consulta y que este se encarga de mostrar, teniendo como resultado final una lista de consultas con los detalles que se quiera mostrar.

La opción de los médicos de nueva consulta simplemente es un formulario con el campo obligatorio de la tarjeta SIP, y con un estado que controla si el medico quiere cancelar o si la consulta se ha guardado correctamente, para redirigirle al inicio.

* 1. **Cita**

Las dos opciones disponibles en el apartado de citas, son listar las citas con la posibilidad de filtrar y nueva cita. Para listar las citas se tienen que hacer tres peticiones a tres métodos diferentes (todas las citas, las de videollamada y las presenciales) según el tipo de usuario. Los resultados de estos métodos hay que mostrarlos cuando se haya cargado el componente, para esto usamos el “componentDidMount()”, para llamar a las funciones que hacen las peticiones y se guardan los resultados en estados que luego mostraremos en el *render.* Por otro lado para controlar el filtrado de las citas se usa un estado que cambia según el *value* del selector, así en el *render* se devuelve un resultado u otro según el valor de este estado.

Para crear una nueva cita, como pasaba con nueva consulta, simplemente es un formulario que si la petición es realizada con éxito devuelve el código de la cita resultante y redirige al inicio. Lo único a resultar en este componente es que se ha usado una librería externa para poder tener el *datepicker* y así, poder elegir la fecha con el calendario.

* 1. **Mensajes**

Para poder listar los mensajes como sucedía en anteriores casos necesitamos los mensajes enviados y recibidos cuando cargue el componente, por eso volvemos a usar el “componentDidMount(), dentro de el se llama a las funciones que hacen la petición para obtener los mensajes y guardar los resultados en estados. Luego en el *render* se listan dichos mensajes llamando a un componente mensaje al que se le pasa los datos de un mensaje, y este se encarga de mostrar los detalles que se considere.

La diferencia con otros componentes está en que en este caso un usuario puede borrar los mensajes enviados o enviar uno nuevo, lo que supone que se tiene que actualizar la lista de mensajes mostrada. Para actualizar la lista y volver a hacer las peticiones se hace uso del “componentDidUpdate(), el cual se llama automáticamente después de una actualización. Como solo queremos que se vuelva a hacer las peticiones tras la actualización de unos casos específicos, se usan estados para controlar estas situaciones.

componentDidUpdate(){

if(this.state.borrado){

this.mensajesRecibidos()

this.mensajesEnviados()

this.setState({borrado:false})

}else if(this.state.enviado){

this.mensajesRecibidos()

this.mensajesEnviados()

this.setState({enviado:false})

}

}

En el componente de crear un nuevo mensaje es un formulario con todos los campos obligatorios, y donde se controlan los estados para cancelar, en el que vuelve a la vista anterior, y para en el caso de que se envíe con éxito un nuevo mensaje, se cambia el estado de enviado para volver a cargar la lista mensajes otra vez.

1. **Conclusiones**
   1. **Revisión de objetivos**

En el proyecto se propusieron diferentes objetivos, el principal era crear un sistema seguro y de código abierto para la comunicación entre medico y paciente sin la necesidad de estar los dos físicamente en el mismo sitio. Proveer funciones adicionales como por ejemplo, crear citas eligiendo el medico y el tipo de cita que se quiera, dar la posibilidad de comunicarse entre ellos mediante mensajes, ver el historial médico y las consultas y poder crear nuevas. Y aparte de todo esto, conseguir que el sistema fuese lo mas seguro posible.

Estos objetivos se han cumplido en su mayoría, teniendo como resultado un proyecto estructurado en dos grandes partes.

En la parte del API REST, se han desarrollado la mayoría de servicios propuestos, exceptuando algunos adicionales como el cifrado de ficheros. Aparte de eso se ha conseguido un API que puede ser reutilizada en otros proyectos, es decir, que no es exclusiva y tiene la posibilidad de que se le añadan nuevas características en el futuro.

En la parte de la aplicación web, se ha conseguido una interfaz de usuario intuitiva pero mejorable en ciertos aspectos, sobretodo de cara a mejorar la experiencia del usuario y reducir el numero de errores que pueda tener. Se pueden añadir funcionalidades que en el API están implementadas pero en el cliente no se han llegado a añadir, como es el caso de editar una cita.

Con respecto a la seguridad, se ha llegado a un nivel de seguridad de datos bueno, cifrando la conexión con los certificados SSL y cifrando los datos antes de almacenarlos con AES256 con CBC. Aun así la seguridad es mejorable, por ejemplo se puede cifrar la base de datos entera no solo los datos, ya que el sistema estaría expuesto ante un posible robo o acceso indebido al servidor. También se puede mejorar el cifrado, en lugar de usar AES con CBC se podría usar AES con GCM, que es otro modo de operación de cifrado en bloque que ha adoptado ampliamente gracias a su eficiencia y rendimiento.

* 1. **Trabajos futuros**

El proyecto tiene muchas posibilidades de mejoras y ampliaciones, como por ejemplo añadir la funcionalidad de subir ficheros adicionales a las consultas como ya se ha mencionado anteriormente.

Al principio del proyecto se planteó también la idea de meter eventos del establecimiento donde este alojada la aplicación, ya sea centro de salud u hospital. Esta característica está contemplada en el esquema relacional de la base de datos (*ilustración 19*), por lo que solo habría que implementar la lógica de negocio en el API y añadir la funcionalidad al cliente.

Otra posible interesante característica es ofrecer al paciente y al medico un calendario con recordatorios de las citas pendientes, y en el caso del paciente también mostrarle los medicamentos que se tiene tomar al día, mandándole un aviso a la hora concreta en la cual se lo tendría que tomar.

Y por supuesto, está también la futura implementación de convertir el sistema a aplicación para dispositivos móviles.

1. **Bibliografía**

NodeJS (2018). *NodeJs.* Obtenido de

<https://nodejs.org/api/>

Monday (2018). *Planificacion.* Obtenido de

<https://monday.com/>

ReacJS (2018). *Componentes.* Obtenido de

https://reactjs.org/docs/react-component.html

React-day-picker (2018). *DatePicker.* Obtenido de

<http://react-day-picker.js.org>

Wikipedia (2018). *Modos de operación (cifrado).* Obtenido de

<https://www.wikipedia.org>

Wikiwand. *Arquitectura cliente-servidor.* Obtenido de

<http://www.wikiwand.com/es/Cliente-servidor>

Murcia Salud (2013). Obtenido de

<https://www.murciasalud.es/principal.php>

Politecnica IUMET. *Telemedicina.* Obtenido de

<http://www.iumet.es/servicios-telemedicina.php>

Vida. *Videoconsultas médicas*. Obtenido de

<https://www.vida.es>

Sanitas (2016). *Telemedicina.* Obtenido de

<https://sanitasventas.com/la-telemedicina-ya-realidad-sanitas-se-llama-blua/>

HealthTap. *Consultorio médico*. Obtenido de

<https://www.healthtap.com/what_we_make/overview>

Atrys. *Telemedicina*. Obtenido de

<http://www.atryshealth.com/es/-qué-es-la-telemedicina-_1580>

We are social (2018). *Global Report*. Obtenido de

<https://digitalreport.wearesocial.com>

Expansión (Enero 2018). *Cifras de internet*. Obtenido de

<http://www.expansion.com/economia-digital/innovacion/2018/02/01/5a72e73a22601db2288b4658.html>

La Vanguardia (Febrero 2018). *El 68% de la población española se conecta a internet a diario*. Obtenido de

<https://www.lavanguardia.com/economia/20180205/44565836290/internet-poblacion.html>

ScienceDirect (Noviembre 2016*). Telemedicina, estado actual*. Obtenido de

<https://www.sciencedirect.com/science/article/pii/S0716864016301195>

Juniper (2017). *Minimum Hardware Requirements for Node Servers and Servers*. Obtenido de

<https://www.juniper.net/documentation/en_US/nfv2.1/topics/reference/ccpe-servers-hardware-spec.html>

Schneier (2005). *Cryptanalysis of SHA-1*. Obtenido de

<https://www.schneier.com/blog/archives/2005/02/cryptanalysis_o.html>