Escuela Politécnica Superior

Sistema seguro para la gestión de datos sensibles

Grado en Ingeniería Informática

Trabajo Fin de Grado

Autor:
Carlos Zamora Sanz

Tutor/es:
Rafael Ignacio Álvarez Sánchez

Junio 2020

Universitat d'Alacant
Universidad de Alicante

Mis mejores agradecimientos:

A mi tutor, Rafa,
por su ayuda,
orientación y
especial dedicación
durante el desarrollo
de este proyecto.

A mi familia,
por estar siempre
animándome
y apoyándome.

Índice

Índice de figuras

1.Introducción

Debido a la rápida expansión del uso de las tecnologías de la información, que cada vez son más usadas y demandadas, también se ha visto incrementada la sensibilidad con respecto a la privacidad de los usuarios. Por ende, la gestión segura que se realice con sus datos es un tema de gran relevancia en la actualidad, llegando a considerarse como un problema crítico en muchos casos.

A modo de ejemplo de casos recientes de robo y tráfico ilegal de datos sanitarios sucedidos en todo el mundo, cabe mencionar que en 2018 la administración sanitaria noruega admitió el acceso no autorizado a datos sanitarios de casi tres millones de ciudadanos debido a un fallo de seguridad en su sitio web [1], o que a principios del año 2019 quedaron expuestos los datos sanitarios de, aproximadamente, dos millones de mujeres chinas debido a un agujero en la base de datos Breed Ready [2]. En Estados Unidos, solo durante el primer trimestre de 2019, se identificaron 96 fugas de datos en el sector sanitarios afectando a, aproximadamente, tres millones y medio de usuarios.

Según un informe de Cesicat [3], el número de datos personales expuestos alrededor del mundo crece año tras año, considerándose que en 2019 unos 2700 millones de datos personales de alrededor del mundo quedaron expuestos, duplicando los 1300 millones que lo fueron el año anterior; y que, aproximadamente, un 30% de los mismos son datos personales de carácter sanitario.

Existen diversos motivos que pueden explicar este tipo de ataques a la privacidad de los usuarios, siendo el económico uno de los fundamentales. En el caso de los datos sanitarios, los ciberdelincuentes pueden crear identidades falsas para estafar o extorsionar a los

individuos o aseguradoras médicas que acaban pagando grandes sumas por recuperar la información que deberían haber protegido.

Algunas de las causas más repetidas que permiten estos ataques son el almacenamiento de las claves sin cifrar o poco complejas lo que, una vez sustraídas, permite su utilización en diversas aplicaciones o el almacenamiento de los datos sensibles en bases de datos no cifradas que permiten relacionarlos con los usuarios.

Debido a la realización de las prácticas de empresa, tuve que trabajar en proyectos en los que se hacía un tratamiento de datos sensibles de carácter médico y constaté que la seguridad estaba lejos de ser un valor añadido en las aplicaciones involucradas. Este hecho, junto a mi inquietud por los temas relacionados con la seguridad de la información, ha motivado la realización de este trabajo.

El proyecto desarrollado ha consistido en el diseño e implantación de un sistema que permite ofrecer un servicio de gestión de datos priorizando la gestión segura de los mismos, atendiendo a criterios de privacidad, integridad y autenticidad de la información, así como de ingeniería del *software*, integridad de servicios o escalabilidad del sistema.

Se ha fundamentado en las bases existentes sobre diseño e implementación de sistemas seguros para la gestión de datos a una escala reducida, dejándolo como un sistema abierto susceptible de futuras ampliaciones de forma natural. Cabe resaltar que el sistema diseñado está enfocado a la gestión de datos estrictamente sanitarios, aunque la idea de diseño es perfectamente adaptable a la gestión de otros tipos de datos.

2.Estado del arte

En los siguientes apartados se describen sucintamente algunos sistemas de tratamiento de datos sensibles, así como normativa al respecto; centrándose, por no hacer demasiado extensa la sección, en el ámbito territorial más cercano. En algún caso se comentan posibles vulnerabilidades que en el sistema desarrollado son franqueadas, también se hace referencia a una comunidad muy extensa de aprendizaje automático ya que en el sistema desarrollado el almacenamiento de ciertos datos se ha despersonalizado, permitiendo así el tratamiento de los mismos para estudios médicos y epidemiológicos preservando la privacidad de los usuarios.

2.1 Reglamento General de Protección de Datos

Las aplicaciones web están obligadas a cumplir con normas legales relacionadas con la privacidad de los usuarios y el tratamiento seguro y responsable de los datos que se ceden. Las aplicaciones relacionadas con el ámbito médico no son una excepción, siendo sistemas que incluso requieren de una protección de los datos más fuerte.

La protección de datos en el ámbito de la medicina es especialmente importante debido al tipo de datos que son tratados. Así, el artículo 6 del Reglamento General de Protección de Datos [4] recoge que son datos de carácter especial los siguientes:

“Datos personales que revelen el origen étnico o racial, las opiniones políticas, las convicciones religiosas o filosóficas, o la afiliación sindical, y el tratamiento de datos genéticos, datos biométricos dirigidos a identificar de manera unívoca a una persona física,

datos relativos a la salud o datos relativos a la vida sexual o la orientación sexual de una persona física."

Cabe destacar que este tipo de datos, que requiere de una protección más activa por parte de la normativa vigente, pueden ser excluidos de esta categoría excepcionalmente en casos muy específicos (como por ejemplo, cuando se da un consentimiento explícito o cuando el tratamiento es necesario por razones de interés público esencial, sobre la base del Derecho nacional o de la UE).

El RGPD también prevé otros derechos que deben ser cumplidos en aplicaciones web como son el derecho de acceso a los datos del usuario, derecho a solicitar la rectificación o supresión de los datos del usuario, bajo determinadas circunstancias; la limitación del tratamiento de los datos del usuario; derecho a la portabilidad de los datos del usuario; y el derecho de oposición al tratamiento de los datos del usuario.

2.2 Abucasis II

Abucasis II [5] es un sistema, implantado en la Comunidad Valenciana, que tiene como objetivo la informatización de la asistencia ambulatoria en atención primaria (Centros de Salud, Unidades de Apoyo de Atención Primaria y Consultorios Auxiliares) y especializada (Centros de Especialidades y Consultas Externas de hospitales). El sistema conecta los centros de forma integral, soportando funciones administrativas (citación, gestión de agendas, etc.) y, ante todo, la gestión del proceso asistencial completo. Toda la información clínica y administrativa del paciente está centralizada y disponible desde cualquier punto de la red asistencial pública (consultorios, centros de salud, centros de especialidades y hospitales), mediante la historia clínica informatizada única por paciente.

El sistema permite la informatización de una gran parte de las labores desempeñadas por el personal médico y sanitario, por lo que debe asegurar la validez jurídica de los documentos implicados, así como la fiabilidad de las transacciones electrónicas que se produzcan. Para asegurar este proceso la aplicación se apoya en certificados digitales emitidos por la Autoridad de Certificación de la Comunidad Valenciana (ACCV) y utiliza firma electrónica avanzada a la hora de realizar actuaciones médicas. Los certificados utilizados garantizan los niveles de seguridad requeridos.

Abucasis-II es utilizado por un número elevado de usuarios (en la actualidad más de 10.000 usuarios), encontrándose todos ellos dispersos a lo largo de toda la Comunidad Valenciana.

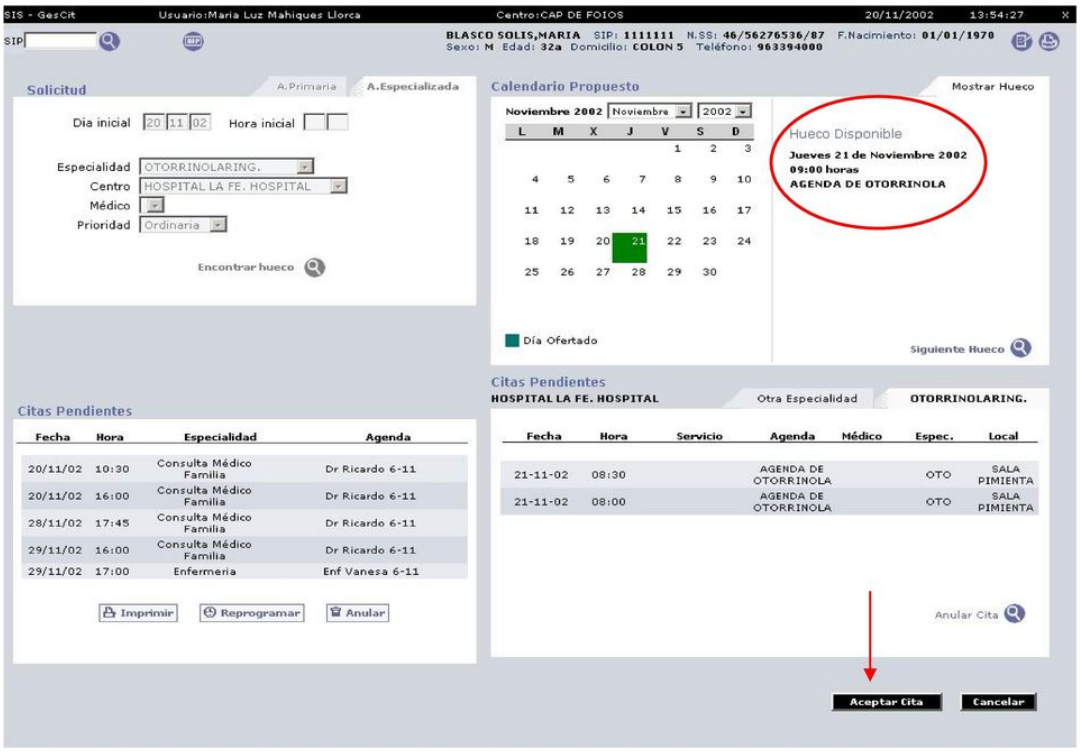

Figura 1.- Interfaz gráfica de Abucasis II

Por esta razón la administración responsable consideró que la opción más eficiente para el correcto desempeño del sistema era establecer una serie de puntos de registro de usuarios (PRUs) propios, que son puntos desde los que se puede solicitar un certificado digital, y que se encuentran distribuidos a lo largo de las distintas áreas de Salud de la Comunidad Valenciana con el fin de prestar servicio exclusivo a personal médico y sanitario. Se ha planificado el despliegue de 40 PRUs ubicados en las Unidades de Gestión de las distintas Áreas de Salud.

La aplicación, cuya interfaz se muestra en la figura 1, se caracteriza por la rapidez con la que un profesional puede tener acceso a un certificado digital, con el objetivo mantener la calidad de asistencia al paciente. La existencia de los PRUs propios de Sanidad permite el despacho urgente de certificados en situaciones de extrema necesidad en un tiempo máximo de dos horas en todo el tramo horario asistencial (24 horas, 7 días a la semana). Esta respuesta se extiende también a la gestión segura y rápida de las contraseñas de acceso a los certificados digitales en caso de pérdidas o bloqueos.

En la puesta en funcionamiento del proyecto, la Conselleria cuenta con la infraestructura y la experiencia de la ACCV, que pone a su disposición su infraestructura, los sistemas informáticos para la tramitación y gestión de certificados digitales, el soporte técnico al

personal encargado de la emisión de certificados y el primer nivel de soporte técnico a los usuarios de la aplicación, siempre que la problemática esté asociada al uso de los certificados.

Adicionalmente, Abucasis II cuenta con un sistema de archivo de registros de acceso, obligando a dar un motivo para todo profesional que solicita la historia clínica de un paciente antes de abrirla. El sistema también obliga al usuario a dejar un rastro que puede ser seguido, permitiendo realizar procesos de auditoría con relativa sencillez.

2.3 IMED Hospitales Online

IMED Hospitales Online es un portal web [6] pionero en la implantación de la historia clínica de pacientes de forma electrónica dentro del marco de la Comunidad Valenciana. El grupo ha adoptado una política de "hospital sin papel", filosofía que llevan aplicando desde hace más de 15 años. En 2016 IMED ganó el premio "Mejor uso de la tecnología en turismo sanitario" otorgado por IMTJ Awards [7].

IMED permite que todos sus clientes tengan acceso a un área privada en la que pueden revisar sus propios informes y pruebas diagnósticas realizadas en sus centros, pudiéndose consultar incluso pruebas que requieren de representaciones gráficas en 3D. Además, tienen la posibilidad de solicitar citas a través de esta plataforma, conociendo en tiempo real la disponibilidad de los doctores. Pese a contar con todas estas funcionalidades, el portal aun presenta, algunos aspectos mejorables desde el punto de vista de la seguridad.

Uno de los aspectos mejorables está relacionado con la simplicidad de la contraseña necesaria para iniciar sesión, que se limita a un código PIN de 4 dígitos. Las contraseñas con una longitud tan reducida aportan una seguridad deficiente, siendo poco complicadas de obtener utilizando ataques de fuerza bruta. Para hacer el sistema más seguro, es recomendable utilizar contraseñas de al menos 8 caracteres, conteniendo mayúsculas, minúsculas, números y caracteres especiales.

El sistema de comunicación con clientes es otro aspecto a mejorar, ya que en ocasiones se envía información sensible a través de canales de comunicación poco seguros, como correos electrónicos. Además el sistema permite que cualquier empleado del centro pueda acceder a todo el expediente del paciente, sin realizar discriminación por tipos de datos o criterios que establezca el propio paciente. El proyecto realizado viene a solventar este último problema,

implementando un sistema de acceso granular, donde se puede establecer a qué porciones del expediente tendrá acceso un empleado.

2.4 Kaggle

Kaggle [8] es una comunidad *online* de científicos de datos y profesionales del aprendizaje automático. Kaggle permite encontrar y publicar conjuntos de datos, explorar y construir modelos de datos científicos, trabajar con otros científicos y profesionales del aprendizaje automático e inscribirse en competiciones de resolución de problemas relacionados con el aprendizaje automático. Kaggle hace uso de más de 19000 *datasets* públicos y 200000 *notebooks* públicos, abarcando una gran cantidad de información.

Kaggle es conocido por la publicación de competiciones con desafíos de aprendizaje automático a gran escala que permiten predecir resultados a problemas muy complejos, generalmente con fines comerciales. Entre las competiciones más destacadas podemos encontrar:

- **Jigsaw Toxic Comment Classification Challenge**: concurso que propone predecir la existencia y tipo de comentarios tóxicos en Wikipedia.
- **Zillow Prize** : se propone la creación de un algoritmo de aprendizaje automático que pueda desafiar a Zestimates, el algoritmo de estimación de precios inmobiliarios de Zillow.
- **Allstate Claim Prediction Challenge**: se propone predecir qué póliza de seguro comprarán los clientes en función de su historial de compras.

Los concursos atraen a expertos de todo el mundo y se ofrecen cuantiosos premios que pueden llegar a alcanzar el millón de dólares. Estos concursos son accesibles para todo tipo de personas, pudiendo participar desde una persona sin experiencia a un completo experto, siendo el concurso una ventana para aprender y mejorar en el campo del aprendizaje automático.

Kaggle también es usado en el ámbito de la investigación, existiendo concursos dedicados únicamente a experimentar con ciertos problemas. Por ejemplo, algunos de los concursos de investigación más destacados han sido:

- **Google Landmark Retrieval Challenge**: dada una imagen encontrar los mismos puntos de referencia en un conjunto de datos.

- **Large Scale Hierarchical Text Classification**: se propone clasificar documentos de Wikipedia en más de 300000 categorías.
- **Right Whale Recognition**: se propone identificar ballenas en peligro de extinción a partir de fotografías aéreas.

Actualmente la comunidad Kaggle realiza contribuciones útiles para la lucha contra el COVID-19, realizado estudios mediante algoritmos de aprendizaje automático que tratan de resolver preguntas abiertas acerca de esta enfermedad. Se puede destacar un concurso que trata de realizar una previsión de casos y muertes a causa del COVID-19 y que puede aportar mucha información al campo de la epidemiología.

Otro concurso propuesto por una aseguradora médica norteamericana, también relacionado con datos médicos, trata de responder a preguntas como qué hospitales tienen mejor calidad de atención o cuál tiene unos costes promedio más altos. La aseguradora, Medicare, aporta al concurso *datasets* con datos anónimos de sus asegurados, siendo usados en procesos de Big Query por los concursantes.

Existe otro concurso propuesto también por una aseguradora médica norteamericana, en este caso Heritage Provider Network que, tras haber realizado estudios, concluye que cada año se desperdicia una gran cantidad de dinero y recursos debido a hospitalizaciones innecesarias. El concurso propone realizar un algoritmo que determine si un paciente requiere de hospitalización en función de todos los datos disponibles acerca de él. Dicho algoritmo también debe predecir cuántos días del año puede pasar cada paciente en el hospital, permitiendo a los centros médicos diseñar nuevas estrategias de trabajo que permitan reducir costes innecesarios. El premio otorgado por lograr dicha meta puede llegar hasta los 3 millones de dólares.

Kaggle es un buen ejemplo para resaltar que las áreas de aprendizaje automático y Big Data están muy al alza hoy en día, siendo campos de los que se puede obtener mucho rédito, científico y económico. Por esta razón, cada vez es más importante obtener conjuntos de datos amplios, estructurados y bien definidos.

3.Planificación

En este apartado se procede a detallar la metodología de trabajo empleada para el desarrollo del proyecto, así como una especificación aproximada del proceso de implantación del proyecto en un entorno de producción realista.

3.1 Metodología de desarrollo

La metodología de desarrollo empleada en el proyecto ha tratado de seguir, en líneas generales, los principios de la metodología ágil XP (Extreme Programming o Programación Extrema). Esta metodología se imparte en asignaturas como AESS (Análisis y Especificación de Sistemas Software) o GPI (Gestión de Proyectos Informáticos) del Grado en Ing. Informática.

La metodología XP pone especial énfasis en potenciar la retroalimentación entre cliente y equipo de desarrollo. En el proyecto podemos considerar que se ha producido esta retroalimentación entre tutor y alumno. Además, es una metodología idónea para proyectos con requisitos cambiantes y poco precisos.

En XP se producen una serie de iteraciones en las que se marcan objetivos, designando unas fechas de entrega que no siempre se cumplen de forma precisa, pero que sirven de orientación. En el proyecto se ha dado esta situación muy a menudo, ya que desde su comienzo se han ido concertando tutorías en las que se definían una serie de objetivos y se programaba una fecha de cumplimiento. La periodicidad de estas reuniones ha oscilado de 1 a 3 semanas, dependiendo de la carga de trabajo. En la figura 2 se ilustra el diagrama de explicativo de la metodología empleada para el desarrollo del proyecto.

Figura 2.- Diagrama de la metodología de desarrollo empleada en el proyecto.

Cabe destacar que la última fase de desarrollo del proyecto se ha visto alterada por una circunstancia excepcional: la creciente difusión del COVID-19 y la consecuente cuarentena sufrida en toda España. Esta situación ha alterado, de forma muy leve, la metodología que se seguía hasta entonces; teniendo que modificar, principalmente, la forma de realizar las reuniones entre tutor y alumno.

Debido a los cambios en la movilidad se sustituyeron las tutorías presenciales por tutorías *online*, concertándolas a través de correo electrónico con una frecuencia de entre 1 y 2 semanas. En estas reuniones se mostraba el trabajo implementado en la iteración anterior, se marcaban objetivos con respecto a la siguiente iteración y se resolvían dudas acerca de cuestiones relacionadas con el desarrollo del proyecto. Una ventaja del teletrabajo ha sido el hecho de poder compartir pantalla a la hora de mostrar las novedades implementadas en cada iteración, agilizando la comunicación entre tutor y alumno notablemente.

El sistema de gestión de proyectos empleado para organizar las tareas que han ido surgiendo durante el proceso de desarrollo ha sido Trello [9]. Este sistema consta de un tablón en el que se pueden añadir columnas, que en el caso del proyecto representan listas independientes nombradas con los diferentes estados que puede tener una tarea (por hacer, en implementación, en prueba y desarrollado). Dentro de cada lista, se han añadido tarjetas con una breve descripción de la tarea y un color que indica la prioridad de esta. En la figura 3 se muestra el tablón Trello empleado a la hora de organizar las tareas a realizar.

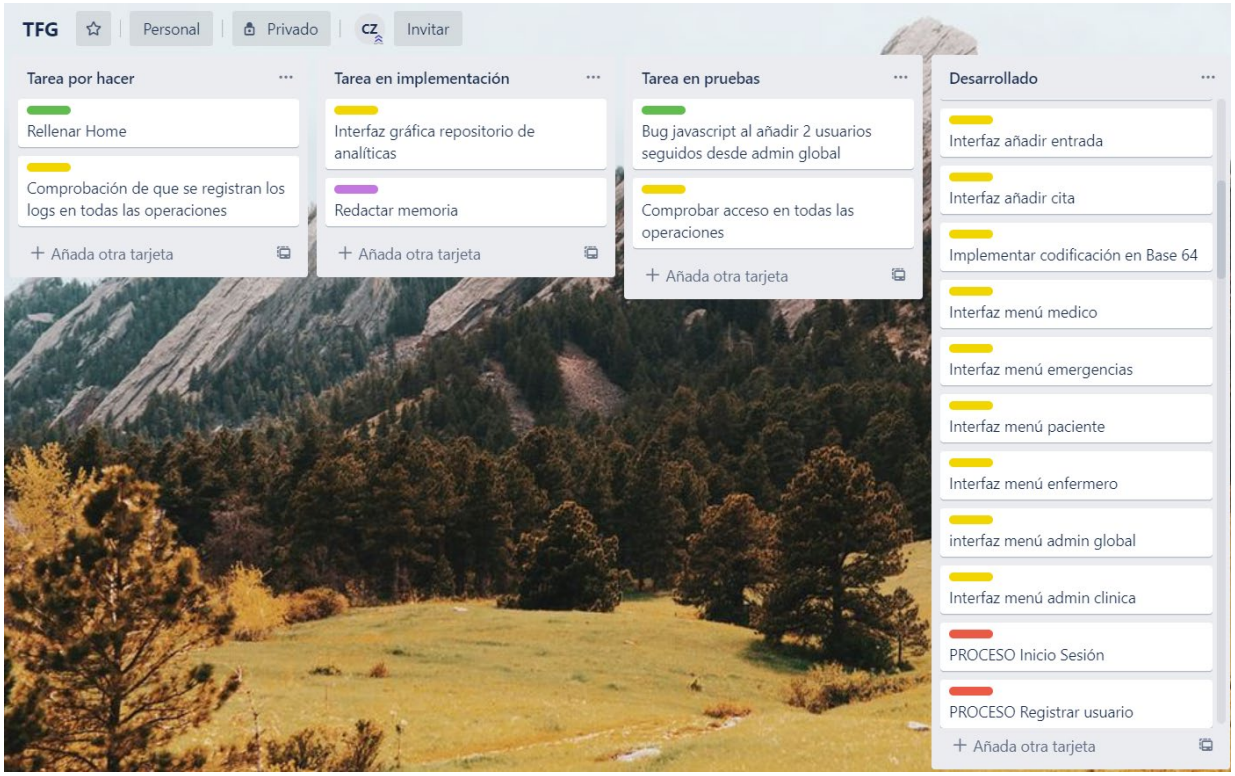

Figura 3.- Panel Trello empleado para organizar las tareas del proyecto.

El uso de este sistema de gestión ha sido combinado con la toma de apuntes a mano, ya que una gran parte de las reuniones entre tutor y alumno se han realizado de forma presencial, hecho que hace complicado el registro de tareas en Trello de forma simultánea. Por lo tanto, la mecánica seguida ha consistido principalmente en tomar notas que más tarde han sido plasmadas en el tablón en forma de tareas.

Por otra parte, se ha utilizado Github [10] como sistema de control de versiones para el proyecto [11]. Github es una herramienta gratuita y extensamente conocida que ya ha sido utilizada en multitud de asignaturas a lo largo del Grado en Ing. Informática, siendo su uso relativamente sencillo. La mecánica de trabajo empleada ha consistido en realizar *commits* cada vez que se termina de implementar una funcionalidad nueva, estableciendo así puntos de restauración definidos de una forma clara y concisa. En ningún momento se ha necesitado volver a un punto de restauración, pero haber dejado puntos bien definidos garantiza cierta seguridad a la hora de programar. En la figura 4 se muestra el repositorio empleado para el desarrollo del sistema.

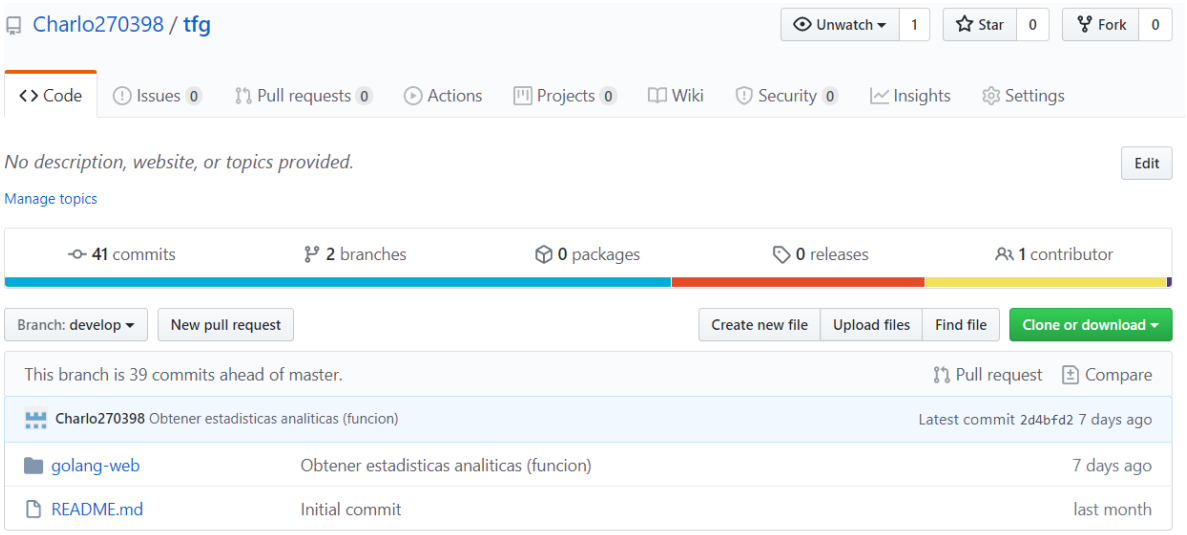

Figura 4.- Repositorio Github utilizado en el desarrollo del proyecto

3.2 Implantación del sistema

Se ha implementado un prototipo del sistema, que actúa principalmente como prueba de concepto para el modelo propuesto de gestión segura de datos sensibles. Entre otros aspectos, cabe destacar los siguientes:

El *lenguaje de programación Go* que, además de ofrecer una librería robusta de funciones relacionadas con la seguridad, se ha convertido en un estándar de facto [12] para el desarrollo de servicios en la nube, siendo soportado de forma nativa por los proveedores principales como Amazon AWS, Google Cloud o Microsoft Azure. Además, al producir ejecutables estáticos sin dependencias externas, simplifica enormemente las labores de instalación y mantenimiento en entornos distribuidos.

La *arquitectura cliente/servidor* con comunicación mediante HTTPS (véase la sección 4.1), mediante la que se encapsula la gestión segura de los datos en un servidor principal y la funcionalidad del cliente en un servidor intermedio, permitiendo el acceso al sistema por parte de los usuarios finales mediante un navegador web. Esto maximiza la flexibilidad del sistema propuesto tanto en términos de despliegue como de escalabilidad.

El *diseño de datos flexible*, que permite adaptar de forma sencilla el sistema a los distintos casos de uso y flujos de trabajo que puedan surgir. Si bien en el prototipo, por sencillez durante el desarrollo, se hace uso de un sistema de gestión de base de datos relacional común (MySQL), se podría optar por migrar dicha base de datos a un sistema como Amazon Aurora [13] o Google BigQuery [14], que mantienen un modelo de datos relacional pero con una

eficiencia muy superior a otras bases de datos, al ser sistemas creados desde el inicio para su funcionamiento en la nube; si bien cabe destacar que la cesión de la gestión de la base de datos a un tercero puede ser un punto negativo, al depender parte de la seguridad del sistema en agentes externos.

Una posible alternativa consistiría en la integración de un *sistema no relacional* de alta escalabilidad para la gestión de datos en el servidor principal. A pesar de que el diseño de datos (véase la sección 4.2) está basado en un esquema relacional, herramientas como MongoDB [15] permitirían implementar el modelo existente con leves modificaciones.

Un posible modelo de negocio empleado a la hora de comercializar el sistema podría ser la venta de licencias de uso, que deberán ser renovadas dentro de los plazos acordados entre cliente y propietario del *software*. La tarifa base de las licencias sería fija pero, adicionalmente, se puede ofrecer un servicio extra que incluya el mantenimiento del sistema, garantizando la operabilidad y buena calidad del *software* adquirido. Este servicio extra ha de ser regulable, es decir, el cliente puede elegir qué grado de mantenimiento quiere realizar, desde un mantenimiento básico a un mantenimiento exhaustivo. Las tarifas del servicio extra variarán en función del grado de mantenimiento que contrate el cliente. Este modelo puede ser compaginado con la posibilidad de ofrecer modificaciones específicas del sistema para clientes concretos, que deberán ser presupuestadas y desarrolladas siguiendo unos plazos acordados.

Para el desarrollo se optaría por utilizar el sistema de control de versiones GitLab [16], ya que es gratuito y permite su instalación en cualquier servidor, abriendo la posibilidad de ubicar el repositorio en el propio servidor de la empresa, haciendo el proceso de desarrollo más seguro y eficiente si cabe.

El entorno más recomendable para el desarrollo del proyecto es Visual Studio Code [17], ya que es un *software* libre y ampliamente utilizado, con características muy interesantes como la integración de extensiones que facilitan la implementación del código, encontrando extensiones que incorporan funciones de autocompletado y comprobación de errores al codificar en Go, entre otros. El propio Visual Studio Code puede sincronizarse con repositorios de versión de controles, facilitando el proceso integración de código entre varios programadores.

En cuanto a los costes de desarrollo, se deben tener en cuenta aspectos como el tamaño de la plantilla necesaria para llevar a cabo el proyecto, el salario de los empleados contratados,

el alquiler y mantenimiento de la oficina, la compra de materiales y *software* necesario para el desarrollo, etc. Además, si se opta por una solución que implique almacenar la base de datos en la nube, se deberá tener en cuenta el gasto que derive de su uso, que suele ser expresado en dólares por GB por mes; repercutiendo este gasto extra en la tarifa final que se aplique al cliente. Asimismo, se estima que el proyecto requiere de una inversión inicial considerable, que podrá ser amortizada conforme se venda la cantidad de licencias suficientes.

La planificación seguida para el desarrollo y monitorización del proyecto será iterativa e incremental, estando dirigida por casos de uso. El *planning* constará de una serie de hitos o fechas de entrega en las que se realizará una *demo* con el cliente, con tal de realizar pruebas de validación del software implementado. Para cada iteración se creará una lista de tareas que definan el *Sprint* a realizar, permitiendo al equipo de desarrollo autoasignarse las tareas que crean convenientes, fortaleciendo así actitudes como la autoorganización o el trabajo en equipo con el objetivo de compartir conocimiento. Se debe contemplar desde un primer momento la aplicación de políticas que incentiven el teletrabajo desde casa ya que, debido a la situación de pandemia actual, existe la posibilidad de que se vuelva a reducir la movilidad de los ciudadanos; apoyando también esta política debido a la existencia numerosos estudios que avalan la idea de que la productividad de un trabajador es mayor cuando trabaja en entornos que facilitan el teletrabajo.

4.Desarrollo del proyecto

En este apartado procedemos a detallar el diseño del proyecto, haciendo especial énfasis en los aspectos referentes a la seguridad de la información y privacidad de los usuarios.

4.1 Arquitectura del sistema

La arquitectura del sistema consta de tres tipos distintos de nodos: servidor principal, servidor intermedio y cliente Javascript (navegador web), disponiendo de un único servidor principal y pudiendo disponer de múltiples instancias de servidores intermedios y clientes Javascript. El esquema básico se muestra en la figura 5.

En la figura se observa que contamos con un nodo central nombrado servidor principal, que ofrece el servicio de gestión y almacenaje de datos de forma segura, mediante el uso de diversas herramientas criptográficas que extenderemos en el apartado 4.4.

Los servidores intermedios se comunican con el servidor principal utilizando una API, siempre a través de un canal seguro. A su vez hay diversos clientes Javascript que se comunican con un servidor intermedio concreto, también a través de un canal seguro. Cada servidor intermedio adopta el papel de cliente y servidor a la vez, en función de si resuelve una petición a un cliente Javascript o solicita un recurso al servidor principal.

Esta arquitectura nos permite adherir nuevos servidores intermedios de una forma bastante flexible, tal es la necesidad que podría darse en el caso de que una nueva clínica quisiera contar con su propio servidor intermedio para interconectarse con el sistema. El diseño contempla la opción de ubicar nodos intermedios en la nube, permitiendo el acceso al sistema a través de una conexión a Internet.

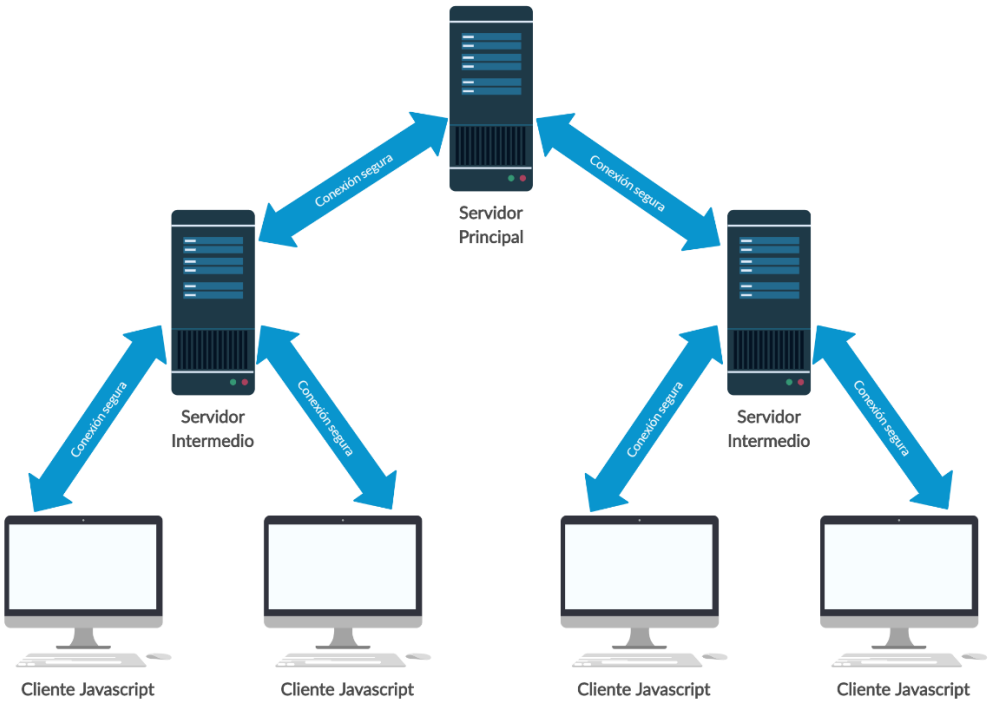

Figura 5.- Esquema básico de la arquitectura del sistema

4.2 Diseño de base de datos

Para garantizar la persistencia de datos se ha optado por el uso de una base de datos relacional, al tratarse de un proyecto en fase de desarrollo y ya que el modelo relacional permite comprender la estructura del esquema de base de datos con relativa sencillez. Además, el modelo de datos relacional garantiza la integridad referencial, de forma que la eliminación de un registro conlleve el borrado de los registros que dependan de él. Esta característica ha sido utilizada en el proyecto, ya que existe una gran cantidad de dependencias en el esquema de base de datos que han de ser controladas. Como se detalla en la sección 3.2, el modelo relacional podría ser sustituido por uno no relacional en caso de que fuera necesario por el volumen de datos o la necesidad de escalabilidad en casos de usos muy específicos.

La base de datos se ubica en la máquina que alberga el servidor principal. La conexión entre el servidor Go y la base de datos se produce a través de un *driver*, que proporciona Go. El *driver* utilizado para la comunicación entre la base de datos y el servidor Go es "Go-MySQL-Driver" [18], permitiendo interactuar con la base de datos de una forma muy similar a la del gestor de base de datos MySQL. El *driver* implementa funciones con las que se pueden realizar consultas o ejecuciones a partir de sentencias SQL con sencillez.

Debido a la gran cantidad de tablas creadas, se ha decidido realizar una partición del diagrama Entidad Relación (ER) que representa la base de datos, agrupando las tablas que tienen un papel en las principales funcionalidades que se han implementado. La partición mantiene como elemento central la tabla Usuarios, ya que la mayoría de las funcionalidades implementadas requieren del uso de esta tabla. También cabe destacar que se puede diferenciar entre tablas auxiliares, creadas por temas de gestión de claves o motivos relacionados con la seguridad; y tablas principales, que recogen las entidades utilizadas en el sistema. Las distintas secciones del diagrama ER se incluyen como apéndice y se describen a continuación.

En la figura 11 se muestran los principales atributos que caracterizan a un usuario, que debe tener como mínimo un rol. Los usuarios con rol de paciente pueden disponer de un historial que, a su vez, puede tener entradas y analíticas asociadas, como se ampliará más adelante. Debe constar cada cita de un paciente, incluyendo el médico que atiende dicha consulta, así como su especialidad y clínica asociadas. Si un usuario adopta un rol que implique un papel de empleado, se insertará su nombre como texto en claro en la tabla “empleados_nombres”.

En la figura 12 se muestran las tablas involucradas en el proceso de solicitud de permiso entre empleados y pacientes. Los empleados autorizados pueden registrar solicitudes de acceso con diferentes niveles de granularidad: acceso total a la historia, a los datos básicos, a una entrada concreta o a una analítica concreta. Para los dos primeros tipos de solicitud, no hace falta guardar un identificador de historial que permita completar el proceso de solicitud, ya que cada paciente tiene un único historial asociado. En los otros dos casos se requiere de identificador del historial e identificador de la entrada o analítica para poder completar el proceso de solicitud.

En la figura 13 se muestran las tablas que intervienen en la compartición de claves de los historiales que contiene el sistema. Se puede distinguir entre tres tipos de entidad a compartir, pudiéndose compartir la clave que cifra los datos básicos de un historial, la clave que cifra una entrada y la clave que cifra una analítica; disponiendo cada tipo de una tabla propia para almacenar las claves compartidas.

En la figura 14 se muestran las tablas involucradas en la gestión de las claves RSA, así como en el proceso de autenticación del usuario. Existe una tabla para almacenar las parejas de claves RSA del sistema y otra para almacenar las parejas de claves RSA de cada usuario. Cada usuario dispone de un *token* con una fecha de expiración que le permite realizar

acciones en el servidor principal, además de un *hash* que resume su DNI con el que el sistema puede realizar búsquedas de usuarios.

En la figura 15 se muestran las tablas que almacenan las analíticas anónimas, cuyos datos se encuentran como texto en claro, siendo totalmente accesibles. Cada analítica tiene un identificador aleatorio que la diferencia de sus homólogas en la tabla que almacena las analíticas del paciente. Además, cada analítica dispone de uno o más tags definidos en otra tabla.

4.3 Sistema de roles

La gestión de los permisos de acceso a la información es llevada a cabo mediante la definición de roles. Los usuarios pueden combinar estos roles; es decir, un usuario puede tener asignado uno o más roles. Pertenecer a un rol implica poseer todos los privilegios asociados a ese rol. De la misma forma, un usuario con más de un rol posee todos los privilegios de los distintos roles a los que pertenece. Los roles son almacenados en una tabla del servidor principal y cada rol dispone de un identificador único, una breve descripción y el nombre que se ha asignado al rol. Cada rol tiene acceso a un menú específico, desde el que se pueden realizar las acciones a las que está autorizado.

La base de datos no tiene en cuenta los roles, ya que toda la lógica de gestión de permisos es realizada por el servidor principal en Go que gestiona la base de datos. Para realizar el control de permisos se ha implementado una función en el servidor principal que, además de comprobar que el usuario es quien dice ser, también verifica que el usuario pertenece a alguno de los roles necesarios para realizar dicha acción.

El sistema implementado consta de los siguientes roles: Paciente, Medicina, Enfermería, Administrador de clínica, Administrador global y Emergencias. Cabe destacar que el sistema permite adherir roles de forma dinámica, pudiéndose añadir más roles en producción sin modificar de forma significativa la estructura del sistema. Dichos roles se describen en las secciones a continuación.

4.3.1 Administrador global

El administrador global tiene la capacidad de modificar entidades dentro del sistema (como usuarios, clínicas, especialidades, etc.), así como listarlas y modificarlas.

Cuando un administrador global añade un nuevo usuario debe rellenar un formulario de registro, en el que se indican datos personales (nombre, apellidos, DNI, dirección de correo,

etc.). Además, en el formulario se debe indicar qué rol o roles va a desempeñar el nuevo usuario. Como ya hemos comentado anteriormente, un usuario debe disponer de al menos un rol, pudiendo combinar varios. Por ejemplo, se puede dar el caso de un paciente que ejerza de médico o un administrador de clínica que ejerza también como personal de emergencias. Adicionalmente, se debe indicar obligatoriamente la clínica en la que va a ejercer un usuario (si se añade un usuario con rol de medicina, enfermería, o administrador de clínica) y a qué especialidad pertenece (en el caso de un usuario con rol de medicina).

Los usuarios con el rol de administrador global únicamente pueden ser añadidos por otro administrador global, siempre desde el formulario anterior. Cabe destacar una excepción puntual, ya que el sistema implementado tiene la peculiaridad de que si el usuario que se va a registrar desde el menú de inicio de la aplicación es el primero en hacerlo (no existen usuarios en la base de datos), este adopta el rol de administrador global automáticamente.

Cuando se registra el primer usuario del sistema, se genera la pareja de claves RSA pública y privada del sistema. Este hecho es muy importante, ya que cada vez que se cifran datos en el sistema se utiliza una clave AES256 y todas estas claves AES256 son cifradas con la clave pública RSA del sistema y almacenadas en la base de datos junto con la información cifrada con esa clave AES256. Por lo tanto, cualquier usuario que disponga de la clave privada RSA del sistema puede descifrar todas las claves AES256 y, en definitiva, descifrar todos los datos del sistema. En la figura 6 se muestra el algoritmo empleado a la hora de crear las claves del sistema, en el caso de registrar un nuevo usuario.

Los usuarios con el rol de administrador global tienen acceso a la clave privada RSA del sistema, al igual que ocurre en el caso de los usuarios con el rol de emergencias. Dicha clave privada RSA del sistema debe ser custodiada de forma segura, detallándose la forma adecuada para hacerlo en la subsección 4.4.6.

Por lo tanto, cuando un administrador global añade a un usuario, bien con el rol de emergencias o con el rol de administrador global, debe recuperar la clave privada RSA del sistema. Una vez recuperada, procede a compartirla con el nuevo usuario aplicando criterios de seguridad, extendiéndose este aspecto en el subapartado 4.4.6. Por lo tanto, la gestión del par de claves RSA del sistema depende desde un primer momento del primer usuario registrado, y posteriormente de todos los usuarios con acceso a ella.

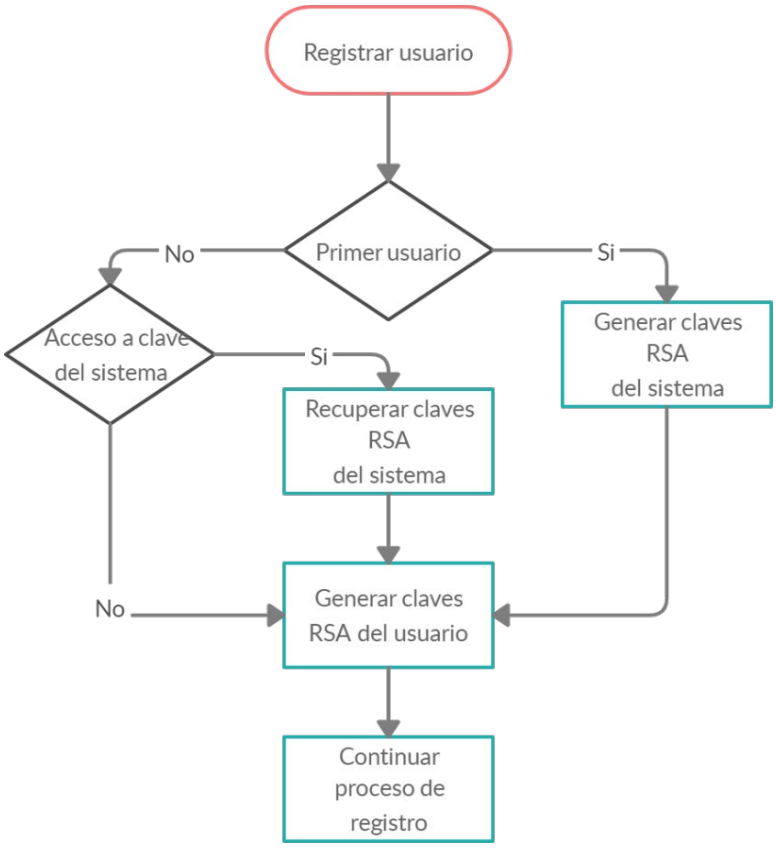

Figura 6.- Algoritmo del proceso de creación de las claves RSA del sistema

4.3.2 Administrador de clínica

Los administradores de clínica tienen la capacidad de editar los atributos relacionados con la clínica, así como añadir nuevos usuarios relacionados con el área de enfermería y medicina, que ejercerán obligatoriamente en la clínica asignada al administrador. También pueden añadir nuevos administradores de clínica a su propia clínica. Además, se dispone de un panel donde consultar información básica de la clínica que se administra (nombre, dirección, tamaño de plantilla, etc.).

Los administradores de clínica únicamente pueden ser dados de alta en el sistema por administradores globales u otros administradores de la misma clínica, rellenando el formulario anteriormente descrito.

4.3.3 Medicina

Los usuarios con el rol del área de medicina tienen la capacidad de realizar consultas sobre las historias clínicas del sistema desde un buscador. En el buscador se debe indicar el DNI asociado a la historia clínica que se quiere consultar.

El facultativo tiene la opción de solicitar el acceso básico a la historia clínica (si no dispone ya de él) o el acceso total. Si ya dispone de acceso a los datos básicos de la historia clínica puede listar las entradas y analíticas que contiene, pudiendo solicitar acceso a las entradas y analíticas individualmente.

Una vez realizada la solicitud, debe esperar a que el paciente autorice el acceso de forma manual. Cuando el facultativo reciba la autorización del paciente, podrá consultar la entrada o analítica. Desde la misma interfaz del buscador también se da la posibilidad de añadir nuevas entradas o analíticas a la historia clínica del paciente, cediendo el permiso de consulta de la entrada al facultativo que realice la inserción forma automática.

Adicionalmente, los usuarios con el rol del área de medicina pueden consultar las citas pendientes desde su menú principal, y cancelarlas si es necesario, además de pasar consulta. Cuando se pasa consulta es posible añadir una entrada a la historia del paciente, especificando el motivo de la consulta y el juicio diagnóstico. En este caso también se otorga de manera automática el permiso de consulta sobre la entrada que ha insertado el facultativo durante dicha consulta.

También es posible realizar consultas sobre las analíticas almacenadas en la base de datos, pudiéndose representar dichos datos en forma de gráficas.

Los usuarios con el rol de medicina pueden ser dados de alta en el sistema por administradores globales o por administradores de clínica, rellenando el formulario oportuno. En el caso de un administrador de clínica, no se debe especificar la clínica en la que va a ejercer el nuevo facultativo, ya que es asignado a la misma clínica en la que ejerce dicho administrador.

4.3.4 Enfermería

Los usuarios con el rol de enfermería tienen la capacidad de consultar historias clínicas desde un buscador en el que se debe indicar el DNI del paciente asociado a esa historia clínica. Como ya hemos explicado anteriormente para el caso del rol de medicina, el usuario con el rol de enfermería también debe solicitar el acceso básico a la historia clínica para poder listar las entradas y analíticas. Si se quiere consultar una entrada o analítica concreta se debe solicitar acceso a la misma. Una vez el paciente dé el visto bueno, el enfermero o enfermera puede acceder a la información. El paciente propietario de esa historia deberá autorizar explícitamente el acceso en cada solicitud que se realice.

A diferencia de los usuarios del rol de medicina, los usuarios del rol de enfermería no tienen la capacidad de gestionar citas con pacientes, ya que no se contempla la posibilidad de concertar citas con enfermería. Tampoco tienen la capacidad de representar gráficas sobre los datos anónimos del repositorio de analíticas, ya que es un privilegio que se reserva únicamente a los usuarios del rol de medicina.

Los usuarios con el rol de enfermería también pueden ser dados de alta en el sistema por administradores globales o por administradores de clínica, rellenando el formulario relevante. Al igual que con otros roles, en el caso de ser dado de alta por un administrador de clínica no es necesario especificar la clínica en la que va a ejercer el nuevo usuario, ya que se asume la misma clínica en la que ejerce dicho administrador.

4.3.5 Emergencias

Los usuarios que disponen del rol de emergencias disponen de una interfaz de búsqueda de historias clínicas similar a la de los roles anteriormente mencionados. Sin embargo, el rol de emergencias dispone de un acceso total a todas las historias clínicas del sistema de forma automática. Esto es posible gracias a que los usuarios con el rol de emergencias tienen acceso a la clave privada RSA del sistema y, por lo tanto, pueden acceder a cualquier historia clínica sin solicitar el acceso.

La implementación de esta funcionalidad viene motivada por la necesidad de que un empleado requiera de la información de un paciente que se persone en una clínica debido a una urgencia o en situaciones de emergencia que así lo requieran.

Los usuarios con el rol de emergencias sólo pueden ser dados de alta en el sistema por administradores globales, ya que se requiere del acceso a la clave privada RSA del sistema. Para añadirlos también se debe rellenar el formulario anteriormente descrito, sin ser necesario indicar una clínica o especialidad.

4.3.6 Paciente

Los usuarios con rol de paciente tienen la posibilidad de concertar citas con facultativos de todas las clínicas del sistema a través de una interfaz. Además, pueden consultar su propia historia clínica de forma completa, pudiendo acceder a las entradas y analíticas que contenga.

También tienen la posibilidad de ceder permisos de acceso a empleados del sistema siempre y cuando exista una solicitud previa formulada por dicho empleado. Esta solicitud podrá ser aceptada o rechazada. Las solicitudes pueden ser de varios tipos, existiendo solicitudes que

implican compartir los datos básicos de la historia clínica, compartir una entrada concreta, compartir una analítica, o compartir la totalidad de la historia clínica. Cuando un paciente recibe una solicitud de acceso, se le muestra una notificación en el menú principal del paciente desde donde se puede acceder al menú de autorizaciones. En el menú de autorizaciones se listan todas las solicitudes pendientes que tiene el paciente.

Cabe destacar que al concertar una cita con un facultativo se le ceden permisos de consulta sobre los datos básicos del historial (nombre, sexo, alergias, etc.) de manera automática.

Los usuarios con el rol de paciente pueden ser dados de alta en el sistema por administradores globales rellenando el formulario anteriormente descrito. También pueden ser dados de alta rellenando el formulario de registro ubicado en la página de inicio, que es accesible por cualquier persona sin ninguna restricción.

En este formulario se deben indicar datos médicos básicos del paciente (sexo y alergias conocidas) y, para validarlo, se deben aceptar los términos y condiciones del sistema. Una vez el paciente es registrado, se añade una nueva fila a la base de datos que contiene la historia clínica del nuevo paciente, donde se incluyen sus datos médicos básicos cifrados.

4.4 Diseño de la capa de seguridad

En los siguientes apartados se describen los diferentes componentes que integran la capa de seguridad del sistema.

4.4.1 Herramientas criptográficas

En este apartado se describen brevemente diversas herramientas criptográficas que ha sido necesario utilizar para gestionar la seguridad en el sistema desarrollado.

Se ha hecho uso de criptografía simétrica para el cifrado de los datos y elegido un algoritmo de cifrado en bloque ampliamente utilizado por ser un estándar con excelentes propiedades de seguridad y eficiencia, el estándar de cifrado avanzado o *Advanced Encryption Standard*, AES [19]. Este algoritmo tiene tres grados de seguridad dependiendo del tamaño de clave elegido: 128, 192 y 256 bits, siendo conocido como AES128, AES192 y AES256, respectivamente. En el sistema desarrollado se ha optado por el nivel más alto: AES256.

La criptografía asimétrica se ha utilizado para el cifrado y compartición de claves, habiendo elegido uno de los algoritmos más utilizados: RSA [20]. A diferencia de AES, RSA requiere de una pareja de claves; una de ellas se debe hacer pública y se utiliza en el proceso de

cifrado mientras que la otra es privada y se utiliza en el proceso de descifrado; esta última se obtiene a partir de la pública conociendo determinados valores, siendo un problema computacionalmente complejo al que se enfrenta cualquier atacante que quiera obtener estos valores y, en consecuencia, la clave privada con la que se descifra.

En el sistema desarrollado, el tamaño de clave elegido para RSA es 2048 bits, considerado seguro para la capacidad de computación actual.

Para garantizar seguridad en las comunicaciones entre usuarios, servidor intermedio y servidor principal, se ha utilizado el protocolo de transporte seguro *Transport Layer Security*, TLS [21], ampliamente utilizado en navegación web, correo electrónico, mensajería instantánea o voz sobre IP.

Una función *hash* permite asignar de forma unívoca una cadena binaria de longitud fija a cualquier información binaria [22]. Se ha generalizado su uso para verificar la integridad de la información y como medida de protección que permite no almacenar las contraseñas de los usuarios en claro, evitando así que puedan ser robadas o accesibles en el sistema. En el sistema desarrollado se ha utilizado la función *hash* SHA3, estándar del NIST, con un resumen de 512 bits, función que hemos representado como $SHA3_{512}$.

Un atacante podría generar una tabla precalculada con una serie de contraseñas y sus correspondientes resúmenes y compararla con la tabla en la que se almacenan los resúmenes de las contraseñas de los usuarios para poder obtener, así, algunas o todas las contraseñas. Para que este precálculo genere tablas muy grandes y la comparación sea muy costosa se utiliza lo que se conoce como sal [23], que consiste en una cadena aleatoria que se concatena con la contraseña elegida por el usuario para hacer el *hash* sobre toda la cadena.

El incremento de la capacidad de computación de las GPU (*Graphics Processing Unit*) ha hecho viable que se puedan comparar todas las contraseñas posibles de un cierto tamaño (lo que se denomina ataque por fuerza bruta) con los resúmenes almacenados en un tiempo muy pequeño. Para evitar este ataque a la autenticación mediante contraseña, se utilizan funciones de derivación de clave basada en contraseña (*Password Based Key Derivation Function* – PBKDF) [24], que son funciones *hash* más lentas, configurables mediante parámetros como la memoria utilizada, el número de hilos y el tiempo de computación. En el sistema desarrollado se ha optado por Argon2id [25], que es una función de derivación de clave seleccionada como ganadora del concurso *Password Hashing* [26]. Esta función incluye la generación de sal para evitar ataques mediante tablas precalculadas.

4.4.2 Canal Seguro

Cuando realicemos conexiones entre los distintos nodos que conforman el sistema desarrollado es muy importante asegurar que la conexión se produce a través de un canal seguro, evitando así posibles ataques como *man in the midle, sniffing,* etc. Facilitar una conexión segura es cada vez más valioso para parte de empresas y entidades diversas, siendo un aspecto crítico en actividades como el comercio *online* o la gestión de bases de datos sensibles.

Para realizar las conexiones seguras entre los distintos nodos que conforman nuestro sistema hemos optado por utilizar la librería crypto/tls de Golang [27], que implementa las versiones 1.2 y 1.3 del protocolo de comunicación seguro TLS (*Transport Layer Security*). Con esta librería ejecutamos una función que, a partir de un certificado y una clave, nos permite establecer un puerto de escucha en el servidor usando dicho protocolo.

En el caso de los servidores intermedios hemos optado por usar el *framework* Gorilla Mux [28], que nos permite realizar enrutamientos de una forma bastante simple. El uso de este *framework* nos permite utilizar la misma función de la librería "crypto/tls" [27] para establecer el puerto de escucha del servidor utilizando el protocolo TLS, pese a que debemos optar por integrarla con el *framework* de una forma algo diferente a la del servidor principal.

Puesto que se trata de un proyecto en fase de desarrollo, hemos optado por el uso de certificados auto firmados. Para generarlos nos hemos servido de la herramienta *openssl* [29], que permite generar certificados auto firmados desde línea de comandos de una manera muy sencilla. Hemos generado un certificado para el servidor principal y otro distinto para el servidor intermedio, siendo deseable generar un certificado distinto por cada servidor intermedio que pretendamos agregar al sistema. De esta forma, cada vez que realicemos una comunicación entre servidor intermedio y servidor principal, el servidor intermedio deberá disponer del certificado y la clave del servidor principal. Ambos residen en la carpeta que alberga el servidor intermedio y se corresponden con los ficheros "cert.pem" y "key.pem", respectivamente.

En resumen, el uso de TLS nos permite realizar una comunicación segura entre los distintos nodos de nuestro sistema, garantizando el intercambio de datos de forma segura y privada entre ellos. De esta forma evitamos la posible sustracción de datos a la hora de realizar peticiones REST de tipo POST, entre otros.

4.4.3 Notación del cifrado

Para simplificar la descripción y facilitar la comprensión de los sistemas de cifrado empleados, se propone la siguiente notación:

$F_s(x)$ = primeros s bits de x
$L_s(x)$ = últimos s bits de x
pwd_{u_i} = contraseña del usuario u_i
pK_{u_i} = clave pública RSA del usuario u_i
pk_{u_i} = clave privada RSA del usuario u_i
Cpk_{u_i} = clave privada RSA del usuario u_i cifrada con AES
pK_s = clave pública RSA del sistema
pk_s = clave privada RSA del sistema
$C_{u_i}pk_s$ = clave privada RSA del sistema cifrada con AES por el usuario u_i
$D_k^a(c)$ = descifrado mediante el algoritmo a con la clave k de la cadena c
$E_k^a(m)$ = cifrado mediante el algoritmo a con la clave k de la cadena m
R = registro descifrado
k_R = clave AES del registro R, por lo general, aleatoria
CR = registro cifrado con AES
h(m) = $SHA3_{512}(m)$ = resultado de aplicar $SHA3_{512}$ a la cadena m
A(m)=Argon2id(m) = resultado de aplicar Argon2id a la cadena m

4.4.4 Registro de los distintos tipos de usuario

Cuando se inicia el sistema desde cero el primer usuario, u_1, va adoptar el rol de administrador global del sistema y se registra en el mismo eligiendo su contraseña, pwd_{u_1}. El servidor intermedio calcula el *hash* SHA3 de 512 bits de esa contraseña, $SHA3_{512}(pwd_{u_1})$, y genera una pareja de claves RSA para el administrador global, pK_{u_1} y pk_{u_1}, y otra pareja de claves RSA para el sistema, pK_s y pk_s. Una vez generadas las dos parejas de claves RSA, las públicas son almacenadas en claro en el servidor principal y las privadas son cifradas en el servidor intermedio con AES256 utilizando como clave de cifrado los 256 últimos bits del *hash* calculado sobre la contraseña del administrador, $L_{256}[h(pwd_{u_1})]$, realizando el siguiente proceso

$$E^{AES256}_{L_{256}\left[h\left(pwd_{u_1}\right)\right]}\left(k_s\right) \quad y \quad E^{AES256}_{L_{256}\left[h\left(pwd_{u_1}\right)\right]}\left(k_{u_1}\right)$$

y siendo enviadas al servidor de base datos donde son almacenadas.

En el formulario de registro el administrador global, además de escribir su contraseña (que debe cumplir requisitos de seguridad) aporta algunos datos personales como DNI, nombre y apellidos, dirección de correo electrónico, etc.; que van a ser almacenados en el servidor principal de una forma particular. El DNI, para garantizar la privacidad, no se almacena en claro si no que se realiza el *hash* $SHA3_{512}\left(DNI_{u_1}\right)$ y el resto de los datos facilitados se almacenan cifrados con AES256 utilizando una clave aleatoria, $k_{R_{u1}}$, generada en el servidor intermedio. Se almacena el hash del DNI, no realizando el cifrado del mismo, porque van a ser necesarias búsquedas en la base de datos utilizando como parámetro el DNI.

La clave aleatoria $k_{R_{u1}}$ es cifrada con la clave pública RSA del usuario, $E^{RSA}_{K_{u1}}\left(k_{R_{u1}}\right)$, siendo almacenada en el servidor principal junto con el resto de los datos facilitados ya cifrados con AES256, $E^{AES256}_{k_{R_{u1}}}\left(R_{u_1}\right)$.

Finalmente, el servidor intermedio envía al servidor principal los primeros 256 bits del *hash* realizado sobre la contraseña del usuario, $F_{256}\left[h\left(pwd_{u_1}\right)\right]$, que serán transformados en el servidor principal para definir la clave de inicio de sesión del usuario u_1, aplicando la función Argon2ID sobre esos primeros 256 bits del *hash* mediante

$$Argon2ID\left(F_{256}\left[h\left(pwd_{u_1}\right)\right]\right).$$

Una vez que el usuario es insertado correctamente se procede a generar un *token* aleatorio, que consta de una cadena de 128 bits La validación de dicho *token* junto con el identificador del usuario asociado a ese *token* es necesaria cada vez que se realiza una acción en el servidor principal. Una vez definida la cadena, se procede a fijar la fecha de expiración o de validez del *token*, limitada a un periodo máximo de 30 minutos desde su generación. Cabe destacar que cada vez que se realiza una nueva acción que requiere de la comprobación del *token* satisfactoriamente, se procede a refrescar otros 30 minutos el tiempo de expiración. De esta forma se evita tener que realizar el proceso de autenticación del usuario de una forma completa cada vez que se sobrepasa el tiempo límite.

Si el proceso de registro se produce sin ningún incidente el *token* e identificador del usuario son enviados desde el servidor principal hasta el servidor intermedio. A continuación se crea un objeto de sesión en el servidor intermedio que contiene el *token* e identificador del usuario. Como ya se ha indicado anteriormente, el *token* y el identificador son requeridos para validar los permisos de ejecución de las peticiones que recibe el servidor principal.

Una tecnología de gestión de *tokens* extensamente utilizada en el contexto de desarrollo actual de aplicaciones web es JSON Web Tokens aun cuando se podría haber utilizado en el proyecto, se ha optado por aportar un sistema de autenticación propio basado en *token*/tiempo consiguiendo gestionar el problema de una forma similar con gran sencillez en la implementación. En la figura 7 se muestra de forma resumida el proceso realizado.

```
//El proceso es realizado en el servidor intermedio

1·Se calcula el hash SHA512 del DNI.
2·Se calcula el hash SHA512 de la contraseña.
3·Se genera el par de claves RSA del usuario.
4·Se genera el par de claves RSA del sistema.
5·Se cifran con AES las claves privadas utilizando los
  últimos 256 bits del hash del paso 1.
6·Se genera una clave aleatoria de 256 bits.
7·Se cifran con AES los datos personales utilizando la
  clave del paso 6.
8·Se cifra la clave del paso 6 con la clave pública
  del usuario, obtenida en el paso 3.
9·Se cifra la clave del paso 6 con la clave pública del
  sistema, obtenida en el paso 4.
10·Se envía al servidor principal los pares de claves
   RSA, los datos cifrados, el hash del paso 1 y la
   primera mitad del hash del paso 2, esperando
   respuesta.
RESPUESTA INVÁLIDA
   11·Se muestra una alerta indicando el motivo del
```

Figura 7.- Algoritmo del proceso de registro para el primer usuario del sistema

En el caso de que un administrador global, como es u_1, dé de alta en el sistema a otro usuario, u_2, con un rol que requiera de la clave privada del sistema (emergencias, administrador global), se seguiría un proceso prácticamente idéntico al anteriormente descrito en el registro del usuario inicial (administrador global, u_1). En este caso, u_1 es quien introduce todos los datos del formulario de registro, indicando adicionalmente qué rol o roles va a desempeñar u_2. Esto quiere decir que u_1 define la contraseña de u_2, pwd_{u_2}, que podrá ser modificada si u_2 así lo desea, mediante un proceso seguro. Este proceso sería más seguro si la contraseña generada fuese aleatoria y conocida únicamente por u_2, pero se ha realizado la implementación priorizando la agilización del proceso de pruebas, siendo muy sencilla y directa su modificación.

Cuando el administrador global, u_1, confirma el registro de un nuevo usuario con el o los roles indicados (emergencias y/o administrador global), el servidor intermedio procede a calcular el *hash* SHA3 de 512 bits de la contraseña introducida, $\text{SHA3}_{512}\left(\text{pwd}_{u_n}\right)$, y genera una pareja de claves RSA para el nuevo usuario, pK_{u_n} y pk_{u_n}. Como el par de claves RSA del sistema, pK_s y pk_s, ya han sido generadas y almacenadas en el servidor principal previamente, u_1 procede a recuperarlas teniendo en cuenta que la pública está almacenada en claro y la privada ha sido cifrada mediante $E^{\text{AES256}}_{L_{256}\left[h\left(\text{pwd}_{u_1}\right)\right]}\left(\text{pk}_s\right)$ por lo que debe realizar

$$D^{\text{AES256}}_{L_{256}\left[h\left(\text{pwd}_{u_1}\right)\right]}\left(E^{\text{AES256}}_{L_{256}\left[h\left(\text{pwd}_{u_1}\right)\right]}\left(\text{pk}_s\right)\right).$$

Una vez recuperadas, pk_s es compartida con el nuevo usuario, u_2, cifrándola con AES256 y clave los 256 últimos bits de la contraseña de u_2

$$E^{\text{AES256}}_{L_{256}\left[h\left(\text{pwd}_{u_2}\right)\right]}\left(\text{pk}_s\right)$$

realizando el mismo procedimiento con la clave privada de u_2

$$E^{\text{AES256}}_{L_{256}\left[h\left(\text{pwd}_{u_2}\right)\right]}\left(\text{pk}_{u_2}\right).$$

El resto del proceso es idéntico al realizado en el registro del usuario u_1. En la figura 8 se resume el proceso realizado cuando un administrador global insertar un nuevo usuario con acceso a la clave del sistema.

```
//El proceso es realizado en el servidor
intermedio

1·Se calcula el hash SHA512 de la contraseña del
  nuevo usuario.
2·Se calcula el hash SHA512 del DNI del nuevo
  usuario.
3·Se genera el par de claves RSA del nuevo
  usuario.
4·Se recupera el par de claves RSA del sistema.
RESPUESTA INVÁLIDA
   5·Se muestra el mensaje de error devuelto.
RESPUESTA VÁLIDA
   5·Se descifra la clave privada del sistema
     usando la segunda mitad del hash SHA512 de
     la contraseña del administrador global.
   6·Se cifran con AES las dos claves privadas,
     utilizando como clave los últimos 256 bits
     del hash de la contraseña del nuevo
     usuario.
   7·Se genera una clave aleatoria de 256 bits.
   8·Se cifran los datos personales del nuevo
     usuario con AES utilizando la clave
     aleatoria del paso 7.
   9·Se cifra la clave aleatoria del paso 7 con
```

Figura 8.- Algoritmo del proceso de registro para los usuarios posteriores al primero que requieran de acceso a la clave privada del sistema

Si el usuario que va a ser registrado es posterior al primer usuario y no requiere de la clave del sistema (roles de medicina, enfermería, administrador de clínica y paciente) el proceso de registro es exactamente el mismo, pasando por alto que pk_s no interviene en ningún momento, por lo que no se realizan los pasos 4 y 5 de la figura 8.

4.4.5 Envío y recuperación de información cifrada

Cuando se introduce información en el sistema se debe diferenciar entre datos sensibles y no sensibles. Los datos no sensibles pueden mantenerse como texto en claro, mientras que los datos sensibles deben ser cifrados obligatoriamente.

El sistema implementado cifra los datos de carácter personal (nombre, apellidos, email, documento de identificación) relativos a cada usuario y los datos médicos de los pacientes (tipos de consulta, datos de analíticas, motivos de consulta, etc.) si bien cabe destacar que existen algunas diferencias en el tratamiento de los datos en función del rol de cada usuario. Por ejemplo, cuando un usuario ejerce como empleado dentro del sistema (roles de medicina, enfermería, administradores, emergencias), el nombre y apellidos son almacenados en una tabla adicional sin cifrar, hecho que se justifica por la necesidad de identificar a los empleados con nombre y apellidos, ya que hay situaciones en las que son necesarios esos datos, tal es el caso de la concertación de una cita por parte de un paciente. Por tanto, se considera que los nombres de los empleados deben ser públicos, planteando como una futura mejora de seguridad que ciertos perfiles de trabajador no se almacenen en claro en el servidor principal.

Otro tratamiento distinto que se puede destacar está relacionado con el DNI de cada usuario. Este dato no es cifrado con AES256, a diferencia del resto de datos personales de un usuario, sino que se almacena el *hash* SHA3-512 junto con un entero que identifica al usuario. De esta forma el DNI no es almacenado en abierto, sino que se almacena una cadena de 512 bits que representa al propio DNI. Cada vez que se realiza una búsqueda se calcula el *hash* sobre el DNI aportado y se compara con las cadenas que se encuentran en la tabla con los *hashes*, obteniendo el usuario asociado al *hash* si existe.

También existe un tratamiento de datos distinto en el caso de los resultados de las analíticas de los pacientes, que son almacenados en abierto. Esta política tiene su justificación debido a que son datos interesantes desde el punto de vista del aprendizaje automático y la investigación médica, por lo que deben poder ser incluidos en un repositorio sin poner en riesgo la privacidad de los pacientes. La solución implementada desvincula al usuario de sus

datos, almacenándolos de forma anónima. En la subsección 4.4.7 se describirá con mayor detalle.

Exceptuando los tres casos anteriormente comentados, el resto de los datos personales del usuario son tratados siguiendo un mismo protocolo. A continuación se describe el seguido cuando un usuario u_3 va a registrarse en el sistema como paciente; se excluyen los pasos ya descritos en el proceso de registro, comentando únicamente el proceso que se aplica a los datos personales y médicos.

En primer lugar u_3 introduce en el formulario sus datos personales, contraseña y datos médicos básicos haciendo uso de un navegador web, siendo enviados a través de un canal seguro (ya descrito en la subsección 0) al servidor intermedio donde serán procesados generando un registro R_{u_3} que será almacenado en el servidor principal realizando el correspondiente proceso de cifrado consistente en la generación de una clave aleatoria $k_{R_{u3}}$ de 256 bits y el uso de AES256: $E_{k_{R_{u3}}}^{AES256}\left(R_{u_3}\right)$.

La clave $k_{R_{u3}}$ es cifrada con la clave pública RSA del usuario, $E_{K_{u3}}^{RSA}\left(k_{R_{u3}}\right)$, siendo cifrada también con la clave pública RSA del sistema, K_s, obteniendo $E_{K_s}^{RSA}\left(k_{R_{u3}}\right)$. La clave pública del sistema, K_s, se encuentra almacenada en el servidor principal debiendo ser recuperada para realizar este paso.

Posteriormente se envían los datos y claves ya cifradas al servidor principal, donde son almacenados en sus respectivas tablas. Los datos personales del paciente son almacenados en la tabla de usuarios, junto con $E_{K_s}^{RSA}\left(k_{R_{u3}}\right)$ y $E_{K_{u3}}^{RSA}\left(k_{R_{u3}}\right)$. Por otro lado, los datos médicos básicos del paciente son almacenados en una tabla con las historias clínicas de todos los pacientes. Los datos médicos básicos son insertados en una nueva fila (cuyo número identificador se corresponde con el del número de historia clínica) junto con $E_{K_s}^{RSA}\left(k_{R_{u3}}\right)$ y $E_{K_{u3}}^{RSA}\left(k_{R_{u3}}\right)$. Paralelamente se realiza el proceso de registro descrito en el apartado anterior.

Cuando el paciente u_3 pretende recuperar sus datos personales o médicos, en primer lugar debe descifrar la clave aleatoria del registro, $k_{R_{u3}}$, almacenada como $E_{K_{u3}}^{RSA}\left(k_{R_{u3}}\right)$, con su

clave privada RSA, k_{u_3}, realizando $D^{RSA}_{k_{u_3}}\left(E^{RSA}_{K_{u_3}}\left(k_{R_{u_3}}\right)\right)$. Como se puede observar, es necesaria la clave privada de u_3, k_{u_3}; para ello se realiza una petición al servidor principal tal y como se detalla en la subsección 4.4.6.

En consecuencia, para recuperar sus datos, u_3 debe descifrarlos realizando el siguiente proceso

$$D^{AES256}_{k_{R_{u3}}}\left[E^{AES}_{k_{R_{u3}}}\left(R_{u_3}\right)\right].$$

Este proceso es similar si cualquier otro usuario autorizado, con acceso a la clave del sistema, necesita acceder a los datos personales de u_3, obteniendo en este caso $k_{R_{u_3}}$ mediante el proceso $D^{RSA}_{k_s}\left(E^{RSA}_{K_s}\left(k_{R_{u_3}}\right)\right)$, lo que supone el siguiente cálculo

$$D^{AES256}_{k_{R_{u3}}}\left[E^{AES}_{k_{R_{u3}}}\left(R_{u_3}\right)\right] = D^{AES256}_{D^{RSA}_{k_s}\left(E^{RSA}_{K_s}\left(k_{R_{u3}}\right)\right)}\left(E^{AES}_{k_{R_{u3}}}\left(R_{u_3}\right)\right).$$

Un aspecto importante a resaltar es la necesidad de recodificar en base 64 [30] aquellos datos (como claves u otros datos binarios) que pudieran presentar problemas de compatibilidad al ser almacenados o transmitidos como texto provocando posibles errores de cifrado y descifrado.

4.4.6 Gestión y almacenamiento de las claves

El sistema implementado requiere del uso de claves para muchos de los procesos que se llevan a cabo, siendo importante la correcta gestión y almacenamiento de las mismas garantizando la seguridad del proceso. Se pueden distinguir distintos tipos de clave que son tratadas en el servidor intermedio y almacenadas en el servidor principal.

Un tipo son las parejas de claves RSA, que se crean en el servidor intermedio y se envían al servidor principal para su almacenamiento, protegiendo la clave privada mediante un cifrado realizado en el servidor intermedio antes de su envío. Este proceso de actuación se repite para todos los usuarios, teniendo que matizarlo para aquellos que tienen un rol que les da acceso a la clave privada del sistema, pk_s, tal es el caso de los nuevos usuarios que adquieran un rol de emergencias o administración global; en este caso la gestión de la pareja de claves personales RSA cuando se registra un nuevo usuario u_n, que adopta ese tipo de rol, consiste en

- la generación de su pareja de claves RSA, pK_{u_n} y pk_{u_n} ,
- el cifrado de su clave privada $Cpk_{u_n} = E^{AES256}_{L_{256}\left[h\left(pwd_{u_n}\right)\right]}\left(pk_{u_n}\right)$ y
- el envío de pK_{u_n} y Cpk_{u_n} al servidor principal para su almacenamiento en una fila de la tabla que contiene las parejas de claves RSA de los usuarios y los identificadores de estos.

Para la pareja de claves del sistema se realiza un proceso similar que consiste en

- la recuperación de la pareja de claves del sistema, pK_s y pk_s ,
- el cifrado de la clave privada del sistema $C_{u_n}pk_s = E^{AES256}_{L_{256}\left[h\left(pwd_{u_n}\right)\right]}\left(pk_s\right)$ y
- el envío de $C_{u_n}pk_s$ al servidor principal para su almacenamiento en una fila de la tabla que contiene las parejas de claves RSA del sistema, pK_s y $C_{u_n}pk_s$, y los identificadores de los usuarios.

Cuando un usuario cualquiera, u_n, requiere del uso de su clave privada realiza una petición al servidor principal que se la suministra, cifrada, si tiene los permisos requeridos. Para poder obtenerla en claro y utilizarla realiza el siguiente proceso

$$pk_{u_n} = D^{AES256}_{L_{256}\left[h\left(pwd_{u_n}\right)\right]}\left(Cpk_{u_n}\right) = D^{AES256}_{L_{256}\left[h\left(pwd_{u_n}\right)\right]}\left[E^{AES256}_{L_{256}\left[h\left(pwd_{u_n}\right)\right]}\left(pk_{u_n}\right)\right]$$

En el caso de que u_n necesite la clave pk_s y tenga los permisos requeridos, el proceso para obtenerla en claro es similar

$$pk_s = D^{AES256}_{L_{256}\left[h\left(pwd_{u_n}\right)\right]}\left(C_{u_n}pk_s\right) = D^{AES256}_{L_{256}\left[h\left(pwd_{u_n}\right)\right]}\left[E^{AES256}_{L_{256}\left[h\left(pwd_{u_n}\right)\right]}\left(pk_s\right)\right].$$

Se observa que, cada vez que se ha de obtener en claro una clave privada se deben utilizar los últimos 256 bits del *hash* $SHA3_{512}$ de la contraseña del usuario, por lo que son almacenados en una variable creada al completar el proceso de registro/inicio de sesión.

En la figura 9 se muestra el proceso realizado para recuperar las claves privadas del usuario y del sistema al iniciar sesión.

```
//El proceso es realizado en el servidor intermedio

1·Se calcula el hash SHA512 del DNI del usuario.
2·Se calcula el hash SHA512 de la contraseña
usuario.
3·Se realiza una petición al servidor principal
  incluyendo el hash del paso 1 y la primera mitad
  del hash del paso 2, con la intención de
  autenticarse; se espera la respuesta.
RESPUESTA INVÁLIDA
   4·Se muestra el mensaje de error devuelto.
RESPUESTA VÁLIDA
   4·Se crea un objeto de sesión, almacenando dentro
     el token devuelto, la identificación devuelta,
     y la segunda mitad del hash del paso 2.
   5·Se solicita la clave privada del usuario y la
     clave privada del sistema para el usuario con
     una petición que incluye el identificador y el
     token del usuario; se espera respuesta.
```

Figura 9.- Algoritmo del proceso de recuperación de las claves privadas al iniciar sesión

Otro tipo de claves que se pueden definir son las claves aleatorias de 256 bits utilizadas en el cifrado AES256 que se realiza sobre los datos del sistema que, como ya se ha explicado anteriormente, son generadas en el servidor intermedio para, posteriormente, ser cifradas con la clave pública del usuario con el que se van a compartir y con la clave pública del sistema, realizándose todo este proceso en el servidor intermedio también. Las claves cifradas son almacenadas en una tabla junto con los datos cifrados, usando una tabla adicional cuando se comparten con otros usuarios.

A modo de ejemplo, a continuación, se describe la gestión realizada con la clave aleatoria AES256 utilizada al cifrar los datos de una entrada del historial clínico de un usuario u_p con rol de paciente para compartir dicha entrada con un usuario u_m con rol médico.

En primer lugar u_p solicita al servidor principal la fila donde se ubica la entrada, recuperando los datos del registro cifrados, $CR_{u_p} = E^{AES256}_{k_{R_{u_p}}}\left(R_{u_p}\right)$ y la clave aleatoria AES256 cifrada con la clave pública de u_p, $E^{RSA}_{pK_{u_p}}\left(k_{R_{u_p}}\right)$.

Seguidamente obtiene su clave privada, pk_{u_p}, enviando una petición al servidor principal, como ya hemos explicado con anterioridad, y realizando

$$pk_{u_p} = D^{AES256}_{L_{256}\left[h\left(pwd_{u_p}\right)\right]}\left(Cpk_{u_p}\right) = D^{AES256}_{L_{256}\left[h\left(pwd_{u_p}\right)\right]}\left[E^{AES256}_{L_{256}\left[h\left(pwd_{u_n}\right)\right]}\left(pk_{u_p}\right)\right]$$

A continuación, u_p recupera la clave aleatoria AES256, $k_{R_{u_p}}$, realizando

$$k_{R_{u_p}} = D^{RSA}_{pk_{u_p}}\left[E^{RSA}_{pK_{u_p}}\left(k_{R_{u_p}}\right)\right].$$

Posteriormente la clave $k_{R_{u_p}}$ es compartida con el médico u_m cifrándola con la clave pública de este. Para ello se solicita la clave pública del médico, pK_{u_m}, y se procede a realizar el cifrado de la clave aleatoria

$$E^{RSA}_{pK_{u_m}}\left(k_{R_{u_p}}\right)$$

siendo enviada al servidor principal, donde se almacenará en una tabla distinta que correlaciona claves compartidas con entradas.

El usuario médico, u_m, estará en condiciones de poder descifrar la entrada R_{u_p}, realizando

$$R_{u_p} = D^{AES256}_{k_{R_{u_p}}}\left(CR_{u_p}\right) = D^{AES256}_{k_{R_{u_p}}}\left[E^{AES256}_{k_{R_{u_p}}}\left(R_{u_p}\right)\right].$$

En la figura 10 se muestra, de forma simplificada, el proceso descrito anteriormente.

```
//El proceso es realizado en el servidor
intermedio
//El paciente ya ha iniciado sesión; Cada petición
requiere el envío del token e identificador del
paciente.

1·Se realiza una petición al servidor principal
  solicitando la clave aleatoria AES de la entrada
  a compartir (cifrada con la clave pública del
  paciente); se espera respuesta.
RESPUESTA INVÁLIDA
   2·Se lanza una excepción.
RESPUESTA VÁLIDA
   2·Se solicita al servidor principal la clave
     privada del paciente.
   RESPUESTA INVÁLIDA
     3·Se lanza una excepción.
   RESPUESTA VÁLIDA
     4·Se descifra con AES la clave privada
       del paso 2 utilizando como clave la segunda
       mitad del hash de la contraseña, que está
       almacenado en el objeto de sesión del
       paciente.
     5·Se descifra con RSA la clave aleatoria del
       paso 1 utilizando la clave del paso 4.
     6·Se solicita al servidor principal la clave
       pública del usuario con quien se va a
```

Figura 10.- Algoritmo del proceso de compartición de la clave de una entrada del historial clínico de un paciente con otro usuario.

4.4.7 Despersonalización de datos

Desde el comienzo del proyecto, se ha valorado la posibilidad de utilizar la información que se almacena en el sistema para procesos de inteligencia artificial, Big Data o investigación. Este planteamiento choca frontalmente con la idea de crear un sistema que almacene la información de forma segura y salvaguardando la privacidad de sus usuarios, ya que se estaría difundiendo información sensible.

La solución adoptada para resolver este problema pasa por almacenar los datos que se pretenden compartir de forma anónima; es decir, son almacenados sin poder ser relacionados con el usuario al que pertenece dicha información. De esta forma, obtenemos un conjunto de datos anónimos de gran utilidad, pero guardando así la privacidad de los usuarios.

El sistema implementado despersonaliza ciertos datos con el fin de poder realizar búsquedas y operaciones sobre ellos, asegurando que no se puedan correlacionar con el usuario del que se extraen estos datos sensibles. Estos datos pueden ser volcados a otra base de datos o a repositorios online, desde los que se pueden realizar procesos de Big Data. La implementación realizada se ha limitado a hacer un pequeño ejemplo de consulta gráfica, dejando la exploración completa de las posibilidades del volcado de estos datos como un trabajo futuro.

En la aplicación, los únicos datos que son despersonalizados son los referentes a las analíticas. Una analítica está compuesta por registros numéricos que representan los niveles de diferentes elementos (como hierro, glucosa, etc.) y se pueden agrupar por etiquetas o *tags*. Los valores de los elementos son de tipo *float*, permitiendo valores numéricos con decimales.

Un *tag* es una categoría que describe una analítica, y ha de ser registrado por un administrador global. Cuando se añade una analítica, se debe indicar al menos un *tag* que pueda describir a esa analítica, permitiendo de añadir más de uno. La existencia de estos tags permite realizar búsquedas parametrizadas como, por ejemplo, comparar la media del nivel de glucosa en pacientes con el *tag* "Hombre" y pacientes con el *tag "*Mujer".

Cuando se registra una analítica, el servidor intermedio cifra los datos y los envía al servidor principal, como ya hemos explicado en apartados anteriores. En el caso de las analíticas anónimas, se envía una petición que únicamente contiene los campos *float* de los elementos y los *tags* añadidos. La petición se procesa en el servidor principal, añadiendo una fila a la tabla donde se almacenan las analíticas anónimas. Cuando se inserta una analítica anónima,

se le asigna un identificador que no se puede relacionar con el usuario ya que, si no, se estaría vulnerando la privacidad del paciente. La forma de llevar a cabo esto pasa por asignarle un identificador aleatorio a la analítica anónima antes de ser insertada. Seguidamente, se procede a insertar la fila de la analítica anónima en la base de datos, generando otro identificador aleatorio si la inserción falla en el improbable caso de que ya exista un identificador igual. Los *tags* asociados a la analítica anónima son insertados en una tabla auxiliar, donde se relacionan analíticas y *tags*.

Los usuarios con el rol de medicina tienen acceso a una interfaz donde pueden operar con estos datos de forma interactiva, pudiendo elegir qué analíticas computar en función de los *tags* que se indiquen. La interfaz ha sido diseñada con D3.js [31].

4.4.8 Librerías utilizadas

En este subapartado se procede a realizar una descripción de las librerías utilizadas en la implementación del proyecto, especificando el/los modos de cifrado utilizados y otras técnicas destacables.

Una de las grandes ventajas del lenguaje con el que se ha implementado la aplicación, Go, es el gran abanico de librerías con funciones criptográficas que forman parte del lenguaje. El uso de estas librerías ha permitido implementar las técnicas de seguridad requeridas para el funcionamiento del sistema propuesto en este proyecto, destacando las siguientes:

- **crypto/rsa** [32]: A partir de las funciones que proporciona esta librería se ha implementado una adaptación de varias funciones RSA. Las funciones adaptadas permiten generar una pareja de claves pública/privada, cifrar una clave AES256 con la clave pública y descifrar una clave AES256 con la clave privada. Adicionalmente se han implementado funciones para transformar las claves RSA a *arrays* de bytes en base 64, que se pueden convertir a cadenas de texto. Es recomendable almacenar las claves RSA en base 64 para evitar problemas de compatibilidad entre las distintas codificaciones.
- **crypto/argon2** [33]: Con esta librería se puede obtener la implementación de Argon2id, utilizando la función IDKey para derivar claves. IDKey requiere de una cadena que indique la contraseña utilizada, un array de bytes que especifique la sal, un entero especificando el tamaño de la sal, y una serie de parámetros que regulen el coste computacional de la operación (como uso de memoria en kibibytes, número de hilos o tiempo). Esta función devuelve un array de bytes como clave. En el manual

de uso de Argon2 se especifica un uso de 64 MB de memoria y que el parámetro de tiempo sea igual a 1 (número de pasadas sobre memoria). También es recomendable obtener una sal aleatoria correctamente, que en el sistema tiene un tamaño de 16 bytes.

- **crypto/rand** [34]**:** Implementa un generador de números aleatorios criptográficamente seguro, que es utilizado en el sistema a la hora de generar las claves aleatorias, los *tokens* o los identificadores de las analíticas anónimas.
- **crypto/aes** [35]**:** Implementa el cifrado AES, anteriormente conocido como Rijndael, de acuerdo con las normas federales del procesamiento de información de EE. UU. Contiene la función NewCipher(key []byte), que crea y devuelve un nuevo cifrador. La función requiere como argumento una clave que debe tener un tamaño 16, 24 o 32 bytes para seleccionar AES128, AES192 o AES256 respectivamente. El modo de cifrado usado en el proyecto es CFB y se indica con la librería "crypto/cipher".
- **crypto/cipher** [36]**:** Implementa modos de cifrado de bloque estándar que pueden ser utilizados en implementaciones de cifrado de bloque de bajo nivel.
- **crypto/sha512** [37]**:** Este paquete implementa los algoritmos hash SHA-384, SHA-512, SHA-512/224 y SHA-512/256 como se define en FIPS 180-4. En el proyecto se usa SHA-512 para generar el *hash* de la contraseña de los usuarios, que es utilizado en otros procesos.
- **encoding/base64** [38]**:** Implementa la codificación en base 64 según se especifica en la RFC 4648. Las funciones de esta librería permiten codificar y decodificar, realizando este proceso antes y después de cifrar todas las claves que gestiona la aplicación.

4.5 Problemas encontrados

Durante el desarrollo del proyecto ha sido necesario solventar una gran cantidad de problemas, en su mayoría debidos al desconocimiento del lenguaje de desarrollo empleado. El uso de Go a la hora de desarrollar el proyecto ha sido, en ocasiones, una dificultad añadida, ya que una gran porción del tiempo destinado a la implementación se ha dedicado a familiarizarse con el lenguaje, ralentizando en ocasiones el desarrollo. Pese a todo esto, la valoración realizada es, en términos generales, positiva ya que haber utilizado Go ha facilitado el trabajo en temas relativos a la seguridad, disponiendo de muchas librerías relacionadas con la criptografía y otros aspectos útiles para el desarrollo del proyecto.

Además, se puede valorar positivamente el haber adquirido experiencia con una tecnología nueva, hecho que será bien recibido en el ámbito laboral.

Otra serie de problemas encontrados son los referentes a aspectos de diseño, principalmente en aspectos relacionados con la seguridad. En su mayoría son debidos a que en el momento de inicio del proyecto no habían sido planteados y, por lo tanto, se han ido resolviendo en fases más tardías del desarrollo. Este tipo de problemas no han resultado ser excesivamente graves ya que se han solventado tomando decisiones adecuadas durante la implementación. Por ejemplo, se decidió adherir el rol de emergencias en una iteración del proyecto sin haber considerado previamente que pueden existir roles con acceso total a las historias clínicas del sistema. Tras plantear el problema, se decidió utilizar la pareja de claves RSA del sistema (que hasta ese momento solo utilizaban los administradores globales) para cifrar con la clave pública todas las claves AES256 del sistema, pudiendo descifrar cualquier dato siempre que se posea la clave privada del sistema. Esto demuestra la flexibilidad del diseño fundamental del sistema, permitiendo ser adaptado a diversas necesidades y casuísticas.

También se han encontrado problemas a la hora de implementar el canal seguro con TLS ya que, al importar los certificados que utiliza el protocolo, se debe tener en cuenta si son certificados auto firmados o no. En el caso de que el certificado no sea auto firmado puede ser añadido sin mayor complicación, pero cuando es auto firmado la librería utilizada exige añadir funciones que validen el certificado. El sistema implementado utiliza certificados auto firmados durante la fase de desarrollo, por lo que se ha tenido que hacer uso de estas funciones. Además, se han detectado problemas a la hora de utilizar el mismo certificado tanto en el servidor intermedio como en el servidor de base de datos, por lo que se ha optado por generar un certificado distinto para cada servidor.

La situación de confinamiento vivida en España también ha sido un factor que ha causado problemas, en gran parte por la incertidumbre que acompañó a las primeras semanas de este. Como ya se ha explicado anteriormente, las medidas de distanciamiento social impuestas por el Gobierno de España han supuesto un cambio en la manera de realizar el seguimiento tutorizado del proyecto, habiéndose solventado mediante la realización tutorías *online*. Cabe destacar que también se ha visto alterada la forma de presentar el proyecto ante el jurado debido a esta situación excepcional.

5.Conclusiones y líneas futuras

La realización del proyecto ha permitido diseñar e implementar un sistema de gestión de datos médicos seguro que supone una aportación al objetivo de garantizar la privacidad de los ciudadanos, particularmente en un tema tan sensible como son los datos sanitarios.

Se ha diseñado e implementado un sistema muy robusto para evitar ataques a la autenticación de usuarios mediante contraseña haciendo uso de técnicas pertenecientes al estado del arte, en particular utilizando una función de derivación de clave, Argon2id, para evitar los ataques más potentes del momento como son los que utilizan granjas de GPU para la obtención de *hashes* o ataques de canal lateral.

Asimismo, se ha fortalecido el tratamiento de las claves necesarias para el cifrado y compartición de la información, diseñando protocolos seguros que involucran el uso de criptografía simétrica y asimétrica; habiendo optado por algoritmos tan solventes como AES256 y RSA con 2048 bits. Todas las claves se almacenan cifradas, en prevención de posibles accesos indebidos.

En el plano de las comunicaciones entre los distintos nodos que conforman el sistema, se ha optado por la utilización de la última versión del protocolo de comunicaciones seguras TLS; garantizando así la confidencialidad, integridad y autenticidad de los datos en tránsito, evitando de este modo posibles suplantaciones de identidad, el conocimiento de los datos enviados y recibidos o la alteración de estos.

De particular interés ha sido, con la despersonalización en el almacenamiento de datos, propiciar el estudio de datos tan interesantes como son los datos sanitarios en un mundo en el que la prevención mediante el estudio de las enfermedades puede ser de especial

relevancia y en el que el aprendizaje automático es una herramienta cada vez más en alza con resultados cada vez más importantes.

Teniendo como objetivo principal la seguridad, en el desarrollo del *software* se ha priorizado la escalabilidad de la arquitectura del sistema por las características específicas del modelo al que está destinado.

El uso del lenguaje Go ha supuesto para la capa de seguridad una garantía y un buen factor de eficiencia, ya que se trata de uno de los lenguajes con librerías criptográficas más completas y específicas.

El presente trabajo, en definitiva, aporta un entorno seguro, reforzado con respecto a otros sistemas existentes en los que se han producido sustracciones de datos de forma masiva.

Si bien se está razonablemente satisfecho con los resultados obtenidos en función de los objetivos inicialmente propuestos cabría hacer algunas ampliaciones para su utilización en un entorno de producción.

Entre otras mejoras se podría ampliar el sistema de *logs* implementado, registrando y clasificando de una forma más exhaustiva los distintos problemas que puedan surgir, como accesos indebidos, errores de inserción, etc.; así como mejorar el sistema de auditoría, de manera que se registren todos los eventos que impliquen acceso, modificación o eliminación de información, realizando un mantenimiento histórico del sistema.

Una ampliación, natural para este trabajo consiste, lógicamente, en la adaptación de la base de datos a las distintas especialidades y pruebas diagnósticas que necesiten ser registradas en el sistema, como documentos, imágenes o representaciones gráficas 3D; dando más sentido, si cabe, a la utilización de Go puesto que para datos con un gran peso la respuesta en el cifrado es bastante eficiente.

También sería muy interesante su ampliación, si se dispusiera de los recursos necesarios, para incluir de forma segura nuevas funcionalidades dentro de la medicina como la telemedicina interactiva, monitorización de pacientes a distancia, almacenamiento y envío con consentimiento de datos clínicos entre centros médicos, etc.

6.Referencias

[1] «Robo Datos Sanitarios Noruega,» 2019. [En línea]. Disponible en: https://www.incibe-cert.es/alerta-temprana/bitacora-ciberseguridad/helsecert-alerta-robo-datos-sanitarios-mas-3-millones.

[2] «Caso de sustracción de datos sanitarios a mujeres en China,» 2019. [En línea]. Disponible en: https://www.theverge.com/2019/3/11/18260816/china-exposed-database-breedready-women.

[3] «Cesicat,» 2020. [En línea]. Disponible en: https://ciberseguretat.gencat.cat/ca/inici.

[4] «Reglamento Genereral de Protección de Datos,» 2020. [En línea]. Disponible en: https://rgpd.es/.

[5] «Manual Abucasis II,» 2020. [En línea]. Disponible en: http://publicaciones.san.gva.es/publicaciones/documentos/V.2164-2003.pdf.

[6] «Imedlevante,» 2020. [En línea]. Disponible en: http://www.imedlevante.com/es/.

[7] «IMTJ Awards,» 2020. [En línea]. Disponible en: https://awards.imtj.com/.

[8] «Kaggle,» 2020. [En línea]. Disponible en: https://www.kaggle.com/.

[9] «Trello,» 2020. [En línea]. Disponible en: https://trello.com/home.

[10] «Github,» 2020. [En línea]. Disponible en: https://github.com/.

[11] C. Z. Sanz, «Github Repositorio TFG,» 5 6 2020. [En línea]. Disponible en: https://github.com/Charlo270398/tfg/tree/develop.

[12] «Go Cloud,» golang.org, 2020. [En línea]. Disponible en: https://go.dev/solutions/cloud/. [Último acceso: 2020].

[13] «Amazon Aurora,» 2020. [En línea]. Disponible en: https://aws.amazon.com/es/rds/aurora/.

[14] «Google BigQuery,» 2020. [En línea]. Disponible en: https://cloud.google.com/bigquery/?utm_source=google&utm_medium=cpc&utm_campaign=emea-gb-all-en-dr-bkws-all-solutions-trial-e-gcp-1008073&utm_content=text-ad-none-any-DEV_c-CRE_335630920203-ADGP_Hybrid+%7C+AW+SEM+%7C+BKWS+~+EXA_1:1_GB_EN_Data+Warehousing_.

[15] «MongoDB,» 2020. [En línea]. Disponible en: https://www.mongodb.com/es.

[16] «GitLab,» 2020. [En línea]. Disponible en: https://about.gitlab.com/.

[17] «Visual Code,» 2020. [En línea]. Disponible en: https://code.visualstudio.com/.

[18] «Go, librería Mysql-Driver,» golang.org, 2020. [En línea]. Disponible en: https://github.com/go-sql-driver/mysql. [Último acceso: 2020].

[19] J. D. a. V. Rijmen, «AES Proposal: Rijndael,» 1999. [En línea]. Disponible en: https://csrc.nist.gov/csrc/media/projects/cryptographic-standards-and-guidelines/documents/aes-development/rijndael-ammended.pdf#page=1.

[20] A. S. a. L. A. R.L. Rivest, «A Method for Obtaining Digital, RSA,» 2020. [En línea]. Disponible en: http://people.csail.mit.edu/rivest/Rsapaper.pdf.

[21] T. Dierks y E. Rescorla, «The Transport Layer Security (TLS) Protocol,» 2008. [En línea]. Disponible en: https://tools.ietf.org/html/rfc5246.

[22] A. J. Menezes, P. C. van Oorschot y S. A. Vanstone, « Handbook of Applied Cryptography,» 1996. [En línea]. Disponible en: https://archive.org/details/handbookofapplie0000mene.

[23] «Salted Password Hashing - Doing it Right,» 2019. [En línea]. Disponible en: https://crackstation.net/hashing-security.htm#salt.

[24] E. B. W. B. a. L. C. Meltem Sönmez Turan, «Recommendation for Password-Based Key,» 2010. [En línea]. Disponible en: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-132.pdf.

[25] D. D. a. D. K. Alex Biryukov, «Argon2: the memory-hard function for password hashing and other,» 2017. [En línea]. Disponible en: https://www.cryptolux.org/images/0/0d/Argon2.pdf.

[26] «Password hashing,» [En línea]. Disponible en: https://password-hashing.net/.

[27] «Go, librería crypto/tls,» golang.org, 2020. [En línea]. Disponible en: https://golang.org/pkg/crypto/tls/. [Último acceso: 2020].

[28] «Gorilla Mux Framework,» 2020. [En línea]. Disponible en: https://www.gorillatoolkit.org/pkg/mux.

[29] «OpenSSL Linux,» 2020. [En línea]. Disponible en: https://wiki.openssl.org/index.php/Command_Line_Utilities.

[30] «What is Base64?,» 2020. [En línea]. Disponible en: https://base64.guru/learn/what-is-base64.

[31] «D3.js,» 2020. [En línea]. Disponible en: https://github.com/d3/d3/wiki.

[32] «Go, librería crypto/rsa,» golang.org, 2020. [En línea]. Disponible en: https://golang.org/pkg/crypto/rsa/. [Último acceso: 2020].

[33] «Go, librería crypto/argon2,» golang.org, 2020. [En línea]. Disponible en: https://godoc.org/golang.org/x/crypto/argon2. [Último acceso: 2020].

[34] «Go, librería crypto/rand,» golang.org, 2020. [En línea]. Disponible en: https://golang.org/pkg/crypto/rand/. [Último acceso: 2020].

[35] «Go, librería crypto/aes,» golang.org, 2020. [En línea]. Disponible en: https://golang.org/pkg/crypto/aes/. [Último acceso: 2020].

[36] «Go, librería crypto/cipher,» golang.org, 2020. [En línea]. Disponible en: https://golang.org/pkg/crypto/cipher. [Último acceso: 2020].

[37] «Go, librería crypto/sha512,» golang.org, 2020. [En línea]. Disponible en: https://golang.org/pkg/crypto/sha512/. [Último acceso: 2020].

[38] «Go, librería encoding/base64,» golang.org, 2020. [En línea]. Disponible en: https://golang.org/pkg/encoding/base64/. [Último acceso: 2020].

[39] «AWS Key Management Service (KMS),» 2020. [En línea]. Disponible en: https://aws.amazon.com/es/kms/.

Apéndice

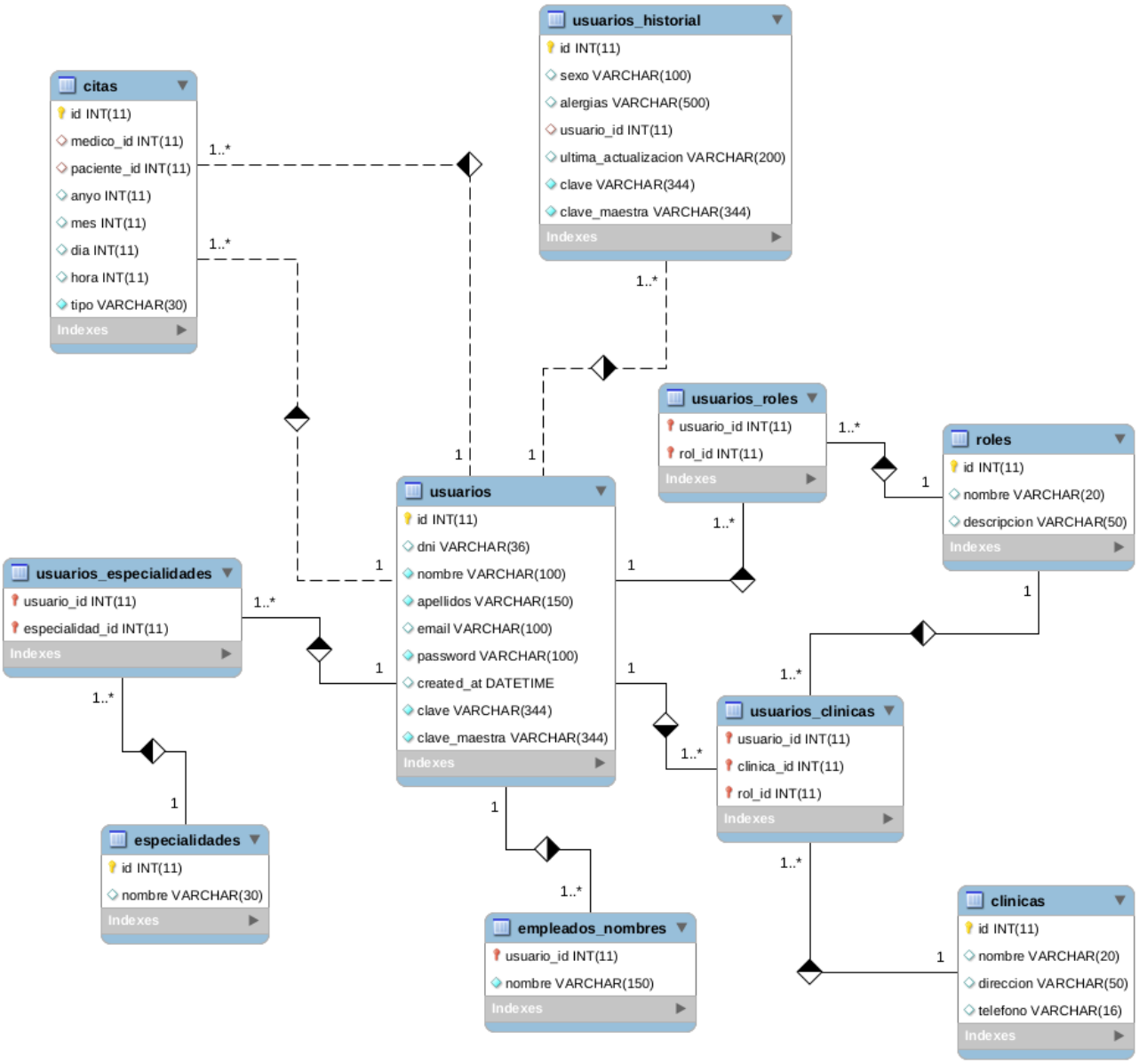

Figura 11.- Diagrama Entidad-Relación de los principales atributos de un usuario

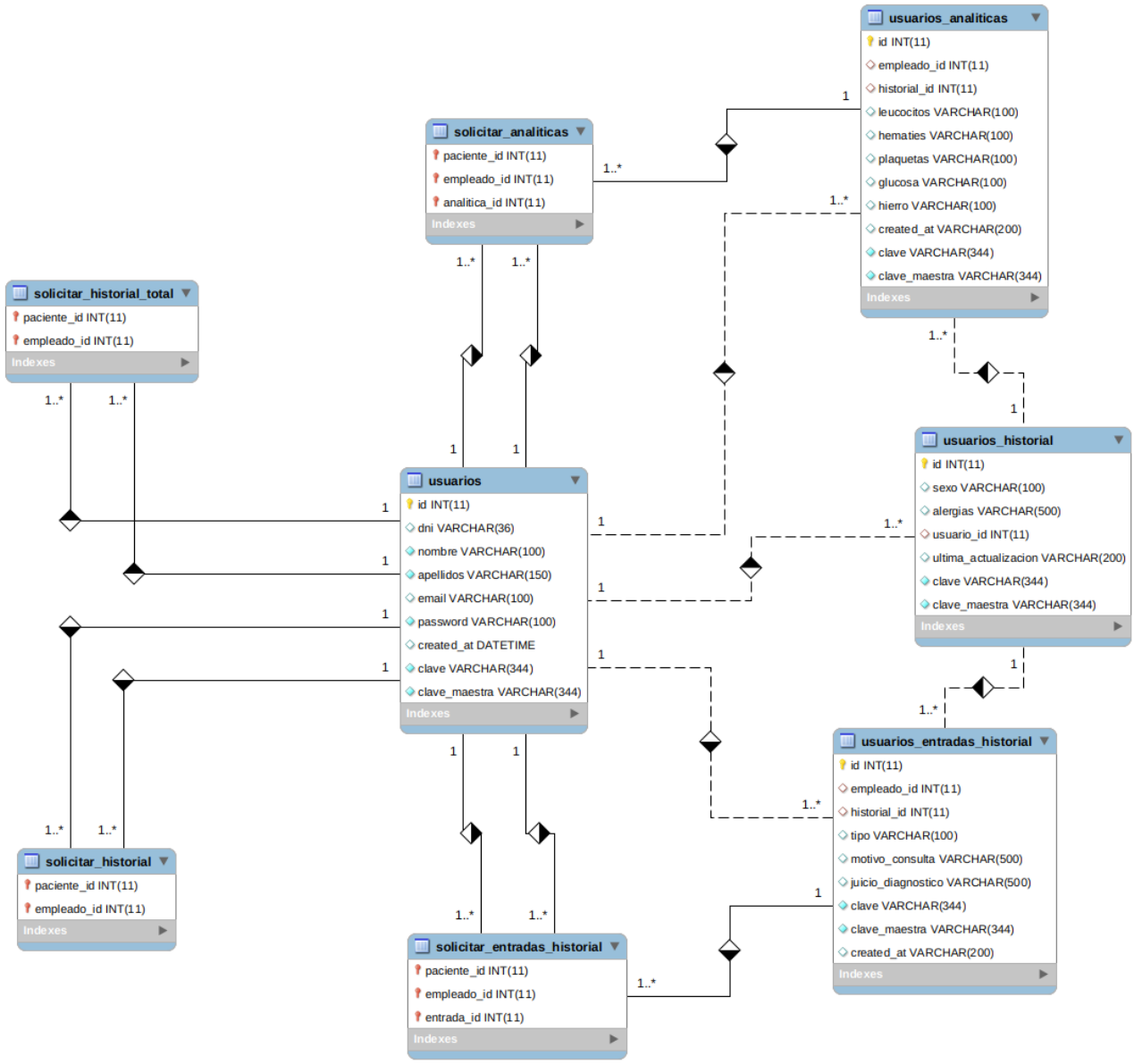

Figura 12.- Diagrama Entidad-Relación de las solicitudes de permisos

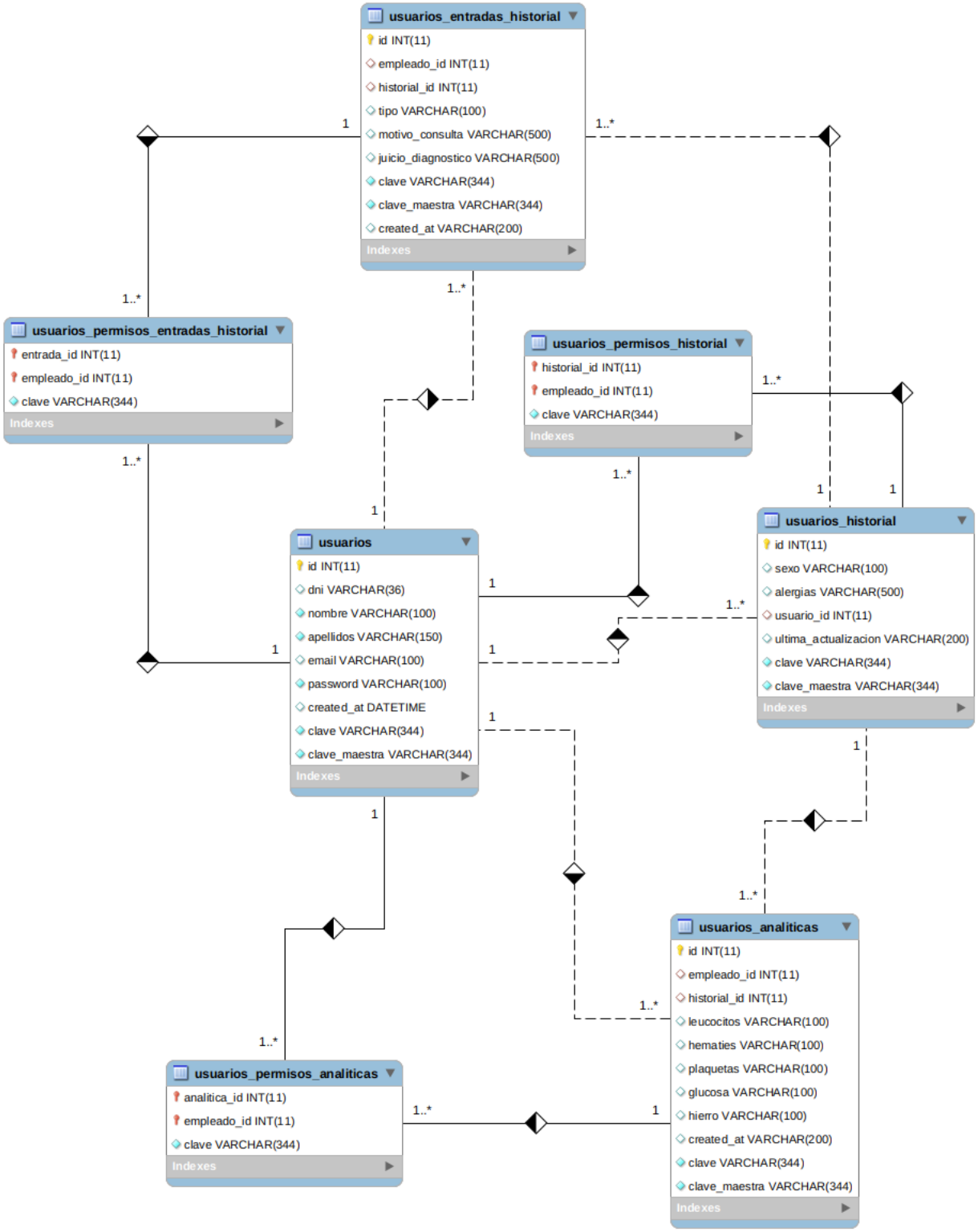

Figura 13.- Diagrama Entidad-Relación del sistema de permisos de la historia clínica

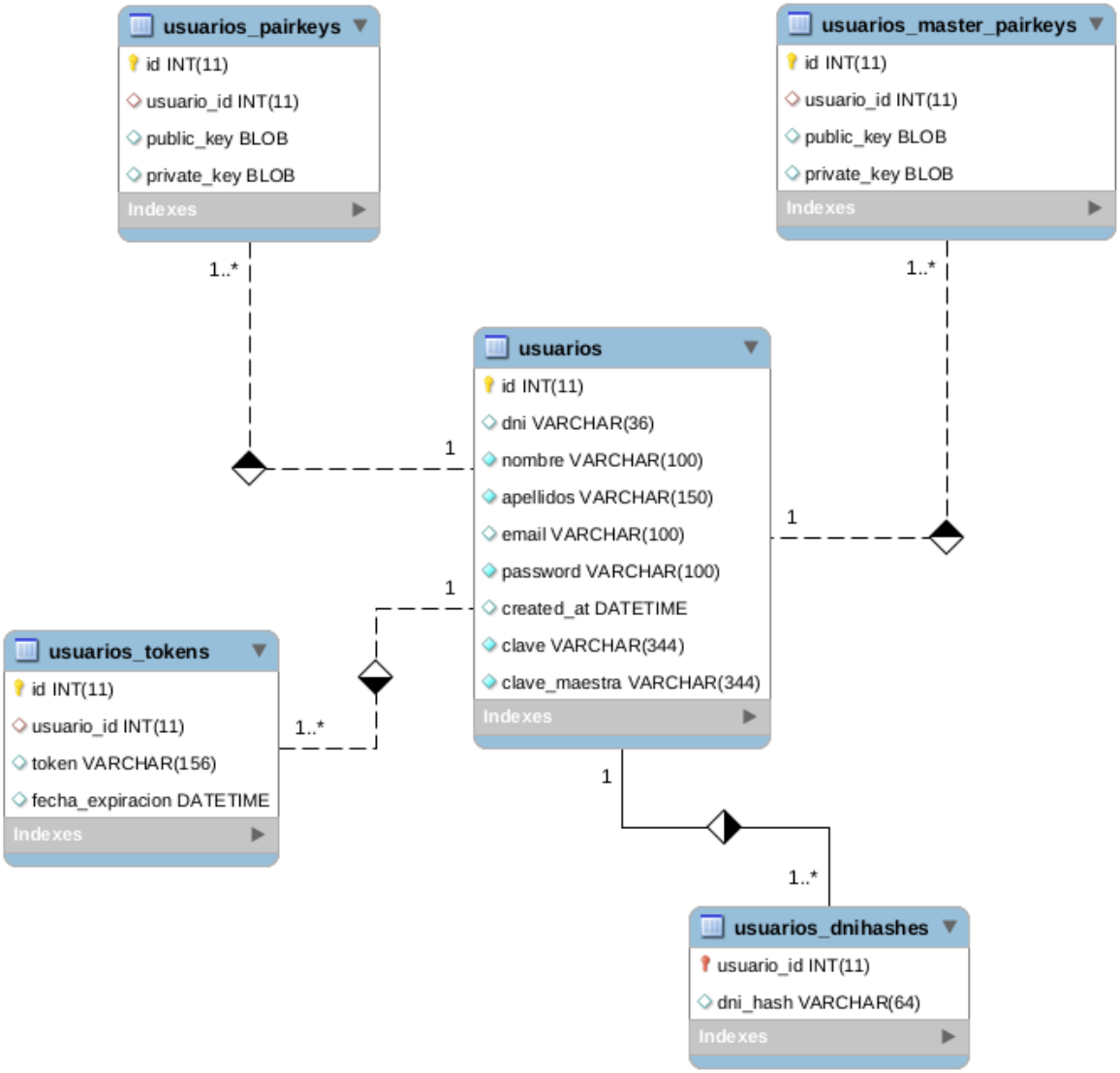

Figura 14.- Diagrama Entidad-Relación de la gestión de claves del usuario

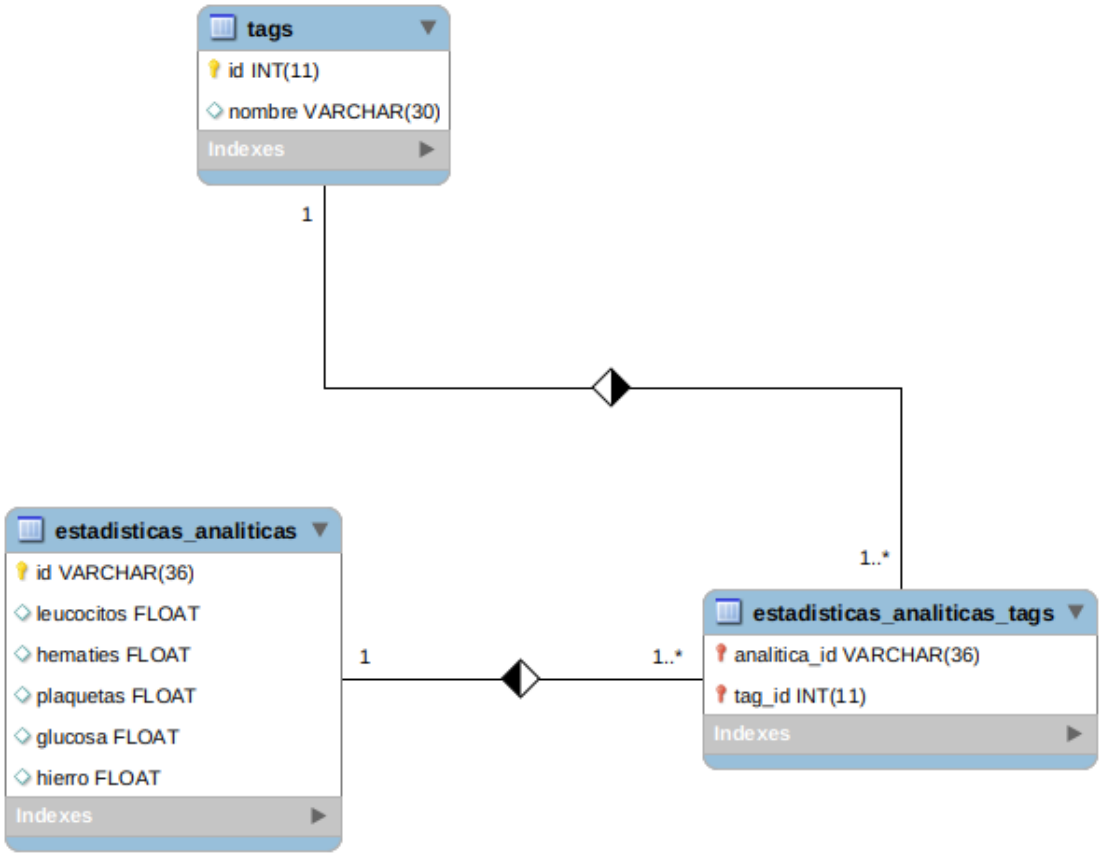

Figura 15.- Diagrama Entidad-Relación de las analíticas anónimas

Se puede consultar el código desarrollado en el siguiente repositorio público [11]:

https://github.com/Charlo270398/tfg/tree/develop

Glosario

PRU: un Punto de Registro de Usuario es un punto desde el que un ciudadano puede solicitar un certificado digital. La solicitud del certificado digital es presencial, siendo obligatoria la identificación del ciudadano mostrando el DNI.

REST: REpresentational State Transfer es un estilo de arquitectura para proporcionar estándares entre sistemas informáticos en la web. Este estilo facilita la comunicación entre los diferentes sistemas. Si un sistema es compatible con REST suele ser llamado RESTful.

POST: es un método utilizado en REST que permite crear un nuevo recurso en la colección de recursos.

RFC: es un conjunto de publicaciones del Grupo de Trabajo de Ingeniería de Internet (IETF en inglés) en las que se describen diversos aspectos del funcionamiento de Internet y otras redes de computadoras.

SSL: Secure Sockets Layer son protocolos criptográficos que permiten realizar comunicaciones seguras en red mediante el uso de criptografía asimétrica y certificados.