UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

HUB DE CITACIÓN ONLINE ENTRE ASEGURADORAS Y AGENDAS MÉDICAS

David de Burgos Fonseca
Tutor: Francisco Javier Perdices Ramírez
Ponente: Miren Idoia Alarcón Rodríguez

MAYO 2019

HUB DE CITACIÓN ONLINE ENTRE ASEGURADORAS Y AGENDAS MÉDICAS

AUTOR: David de Burgos Fonseca
TUTOR: Francisco Javier Perdices Ramírez
PONENTE: Miren Idoia Alarcón Rodríguez

Dpto. Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
mayo de 2019

Resumen

Como Trabajo Fin de Grado se ha desarrollado un proyecto destinado a las aseguradoras médicas, realizado en el entorno laboral para la empresa Redsys Salud. La finalidad de este proyecto es el desarrollo de un sistema que permita consultar la información de las agendas, solicitar y anular citas, obtener horarios y consultar el historial de citas a los asegurados de las compañías, mediante conexiones con diferentes agendas médicas disponibles en el mercado.

El principal problema que se plantea y por el que surge la aplicación es la falta de estandarización y la existencia de múltiples agendas médicas. La aseguradora que desee conectarse con estas agendas necesitaría realizar una integración individual con cada una de ellas. Gracias a este proyecto abstraemos a las aseguradoras de la complejidad de la integración, siendo necesario únicamente realizar una conexión con el sistema del Hub de Citación Online.

El componente principal del sistema es una API REST a la que realizan peticiones las compañías aseguradoras a través de sus propias integraciones. Además, para aquellas compañías que lo deseen, se ofrece un componente secundario, denominado "Modo delegado", el cual se trata de un front-end que consume los servicios de la API, lo que ahorra a las compañías tener que desarrollar un frontal propio.

Se han utilizado tecnologías punteras y metodologías ágiles para el desarrollo del sistema. Para el back-end se ha utilizado Spring Framework con Java EE, utilizando bases de datos relacionales Amazon Aurora, compatible con MySQL. Para el front-end se ha empleado el framework de código abierto Angular, utilizando TypeScript, HTML y CSS.

También se han utilizado tecnologías DevOps para el despliegue de aplicaciones en la nube, en concreto los servicios proporcionados por Amazon Web Services, el PaaS de Openshift, junto a las herramientas y software como Git, Docker, Kubernetes y Jenkins para el control de versiones, la automatización y orquestación de aplicaciones contenerizadas.

Además, se han descrito las metodologías y herramientas de desarrollo ágil empleadas, en concreto el marco de desarrollo ágil SCRUM.

En este documento se ha detallado el proceso del desarrollo de la aplicación. Se han descrito los procesos de análisis y diseño, las metodologías usadas y las tecnologías empleadas en el desarrollo completo de la aplicación.

Palabras clave

Salud, citación online, aseguradora médica, agenda médica, Java, Spring, API, REST, Maven, Apache Tomcat, MySQL, Angular, Angular Material, TypeScript, Amazon Web Services, OpenShift, Docker, Kubernetes, Jenkins, Git, web, Amazon Aurora, Amazon DynamoDB, Amazon EC2, Amazon RDS, SonarQube, Junit, Mockito, Redsys Salud SL.

Abstract

As a Bachelor Thesis, I have collaborated in a project for health insurance companies, that has been done in a work environment for the company Redsys Salud. The goal of this project is the development of a system that allows consulting agendas information, request and cancel appointments, get timetables and appointment history of the insured, through connections with the available agendas in the market.

The main problem that arises and why this application is necessary is the lack of standardization and the existence of a lot of different agenda software. The insurance companies that want to connect to those agendas would need to make an individual integration with each one. Thanks to this project, the complexity of the integration is abstracted from the company, been only necessary one connection with the "Hub de Citacion Online" (online citation hub) system.

The main component of the system is a REST API to which the companies make requests through their own integrations. Also, to those who want, it is offered a secondary component, called "Modo delegado" (delegate mode), which is a front-end that consumes the API services and prevents the companies from developing their own front-end.

It has been used the latest technologies and agile methodologies for the developing of the system. It has been used Spring Framework for the Java EE platform for the developing of the back-end, using relational databases such as Amazon Aurora, which is MySQL compatible. For the front-end, it has been used Angular Framework, with TypeScript, HTML and CSS.

Also, it has been used DevOps practices for the deployment of applications on the cloud, specifically the services provided by Amazon Web Services, OpenShift PaaS, and tools and software such as Git, Docker, Kubernetes and Jenkins for version control, automatization and orchestration of containerized applications.

Moreover, it has been described the different methodologies and technologies for the agile development that have been used in the project, specifically SCRUM framework. In this document, it has been detailed the process of the application development. Also, it has been detailed the analysis and design processes and the different technologies used in the development of the application.

Keywords

Health, online citation, medical insurance, medical agenda, Java, Spring, API, REST, Maven, Apache Tomcat, MySQL, Angular, Angular Material, TypeScript, Amazon Web Services, OpenShift, Docker, Kubernetes, Jenkins, Git, web, Amazon Aurora, Amazon DynamoDB, Amazon EC2, Amazon RDS, SonarQube, Junit, Mockito, Redsys Salud SL.

Agradecimientos

A mi padre, por haberme ayudado a convertirme en la persona que soy ahora, espero que allí donde estés te sientas orgulloso de mi.

A mi madre, por transmitirme toda tu fuerza y enseñarme a seguir siempre hacia adelante.

Porque sin ninguno de vosotros no sería quién soy ahora.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

1 Introducción

En el año 2017 entré como estudiante en prácticas en la empresa Redsys Salud, anteriormente llamada Hava-Soft, dedicada a proveer soluciones tecnológicas al sector salud. Tras un año en la misma se me ofreció la posibilidad de realizar este Trabajo de Fin de Grado a partir del proyecto que estaba desarrollando junto a mi equipo.

Realizar el Trabajo de Fin de Grado en un entorno laboral es una gran oportunidad, que permite mostrar la metodología de trabajo ágil utilizada y el desarrollo de una aplicación real con tecnologías actuales como Angular, Spring Framework, Java EE, los servicios en la nube de AWS (Amazon Web Services), Openshift y tecnologías DevOps como Docker o Jenkins.

Me ha permitido trabajar en varias áreas del desarrollo del proyecto. He participado en el análisis y diseño de la aplicación, analizando los casos de uso, los requisitos y diseñando diferentes diagramas. He desarrollado parte del back-end de la aplicación, incluyendo varios de los servicios web ofrecidos y los módulos de acceso a bases de datos MySQL con JDBC, junto a parte de la batería de pruebas unitarias. He tenido mucha responsabilidad en el desarrollo del front-end con el framework Angular, llevando esta parte casi en exclusividad. Por último, he participado en los despliegues de las aplicaciones sobre los diferentes entornos.

1.1 Motivación del proyecto

La principal razón por la por la que surge este proyecto es la necesidad de las compañías aseguradoras para que sus asegurados puedan pedir citas médicas a través de Internet sobre cualquier médico de su cuadro médico. La digitalización del proceso de reserva de citas es algo muy demandado por los clientes de las aseguradoras privadas, mientras que la sanidad pública ya dispone de este servicio.

El problema surge debido a que cada médico perteneciente al cuadro médico de cada compañía puede utilizar software de agendas distintos, incluso no disponer de uno. En la práctica esto implica que, para poder solicitar cita online, la aseguradora debe realizar una integración individual con N software de agendas distinto. Además, como es común que cada médico trabaje con más de una aseguradora, implica que cada aseguradora debe conectarse con cada agenda disponible en el mercado.

Este es el escenario que la solución del Hub de Citación Online (HCO) pretende cubrir, permitiendo en que, desde cualquier aseguradora, un asegurado pueda pedir cita online sobre cualquier software de agendas médico, idealmente, sobre cualquier médico del cuadro médico de cualquier compañía.

Se ofrece tanto a las aseguradoras como a las agendas médicas un método de centralización, actuando de pasarela y facilitando a las aseguradoras la forma de conectar con cada agenda disponible. El método de conexión se simplifica, siendo únicamente

necesario conectar con el Hub para tener acceso a los servicios ofrecidos por cada agenda del mercado.

Los clientes de las aseguradoras médicas tendrán la posibilidad de solicitar una cita con su médico a través de la página web de su aseguradora, pasando a través de nuestra pasarela, y reflejando en la agenda médica de la que disponga cada facultativo. Además, ofrecerá un servicio de consulta del historial de citas de cada asegurado, permitiendo visualizar a través de una única aplicación toda su actividad médica.

1.2 Objetivos

Este proyecto tiene como finalidad la creación de una plataforma web, cuyo principal componente es una API REST desarrollada en Java EE 8, destinada a ofrecer los diferentes servicios web que consumirán las aseguradoras que quieran conectarse al Hub. A través de esta API se realiza la conexión con las diferentes agendas que estén dadas de alta en el sistema.

Utilizará una base de datos Amazon Aurora, compatible con MySQL, donde se almacenarán los datos de conexión de los diferentes softwares de agendas, la información de los centros médicos, las especialidades médicas, las citas reservadas, junto a otra información. Además, hará uso de una base de datos NoSQL en Amazon DynamoDB, un servicio que ofrece AWS (Amazon Web Services), para almacenar datos de auditoría de forma masiva, en concreto las peticiones y respuestas que se hagan a la API.

Los servicios que ofrece la API están destinados a que las aseguradoras que quieran crear su propio cliente, puedan consumirlos y mostrar las respuestas de la manera que deseen. Como alternativa a las aseguradoras que no deseen implementar por ellas mismas la parte front que consume los servicios, se ofrece la posibilidad de integrarse directamente con un cliente front-end diseñado en este mismo proyecto, y denominado Modo Delegado.

La parte del Modo Delegado está desarrollada con el framework Angular, para el desarrollo de aplicaciones web basadas en el patrón de arquitectura Modelo Vista Controlador (MVC). Hace uso de HTML con etiquetas personalizadas y CSS para mostrar los atributos personalizados dinámicamente. El front-end de Angular realiza peticiones a la parte back-end, que a su vez es la encargada de consumir los servicios de la API de HCO.

Tanto la parte back-end, como la front-end y las bases de datos, se encuentran ubicadas en la nube de Amazon Web Services (AWS), que ofrecen diferentes servicios cloud, como servidores virtuales, almacenamiento y bases de datos, entre otros.

1.3 Metodología y planificación del trabajo

La planificación de este proyecto se ha realizado utilizando la metodología ágil Scrum. Se ha elegido esta metodología de trabajo debido a la complejidad del proyecto, a la necesidad de cambios en los requisitos (a causa de la incorporación de las diferentes aseguradoras y

agendas) y a la necesidad de disponer de entregas continuadas al final de cada iteración para poder ser mostradas a los clientes.

Las metodologías ágiles se empezaron a utilizar en 2001, cuando un grupo de diecisiete desarrolladores de software, cansados de la frustración en la industria de los años noventa, decidieron crear el denominado Manifiesto Ágil, donde se redactaron valores y principios para el desarrollo de software. Surgieron como alternativa a las metodologías tradicionales, más pesadas, lentas y rígidas ante cambios.

Se basan en una estrategia de desarrollo de software que permiten evolucionar los proyectos y adaptar la planificación según las necesidades de cada momento y las condiciones del entorno. Son rápidas y flexibles ante cambios, al realizar mejoras continuas en el producto. Permiten tener un TTM (Time to market) mucho más reducido y un MVP (Minimum Viable Product o Producto Mínimo Viable) mucho antes que con las metodologías tradicionales más pesadas como por ejemplo el desarrollo en cascada.

El marco de desarrollo ágil Scrum [1] está diseñado para equipos con un número reducido de miembros. El trabajo se divide en franjas de tiempo reducidas, denominadas "sprints" (en nuestro caso con una duración de entre dos y tres semanas, variando según las necesidades), que se planifican al inicio de las mismas en lo que se denominan "sprint plannings".

Al inicio del día, se realiza una reunión denominada "daily meeting", de no más de 15 minutos, donde se comprueban las tareas realizadas el día anterior, los bloqueos existentes y se planifica el trabajo que se va a realizar en ese día. Una vez que se finaliza un "sprint", se realiza la revisión y la retrospectiva de este, donde se revisa el trabajo realizado y cada miembro da su opinión y recomendaciones con el fin de mejorar la calidad del proceso y del trabajo.

Existen diferentes roles en Scrum, dependiendo de sus funciones. En primer lugar, el "Product Owner", es el encargado de representar al cliente, ayudando a definir las historias de usuario y la prioridad de éstas; en el caso de nuestra aplicación, debido a que no hay un único cliente, existe la figura del "Proxy Product Owner", que actúa como intermediario entre las personas que toman las decisiones y el equipo de desarrollo. Por otro lado, el "Scrum Master" es el encargado de supervisar que la metodología se está aplicando correctamente y de eliminar los obstáculos que dificultan la aplicación de la misma. Por último, el "Development Team" o "Equipo de Desarrollo" está formado por los miembros que se encargan de desarrollar y entregar el producto.

Para la organización y planificación de los proyectos se anotan las tareas y los estados en los que se encuentra cada una en un tablero. Para agilizar y mejorar este proceso se ha utilizado la plataforma Taiga [2], que proporciona las herramientas necesarias de forma online, como el tablero, historias de usuario, gráficas con la carga de trabajo restante, entre otras.

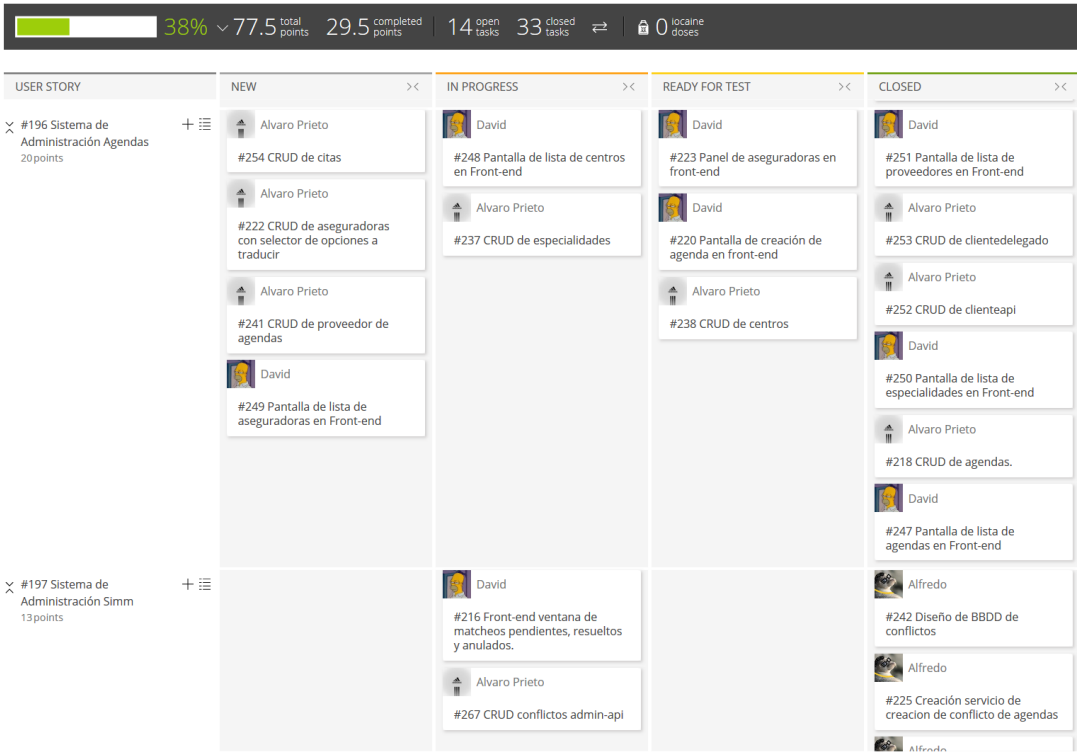

Ilustración 1: Tablero de tareas de Taiga

1.4 Organización de la memoria

La memoria está dividida en varias secciones que hacen la lectura más amena, tocando todos los puntos del desarrollo del proyecto.

En primer lugar, se trata el apartado de “Estado del arte”, en el cual se hablan del estado actual del sector sanitario, la citación online y las tecnologías que están presentes.

A continuación, en el apartado de “Análisis y diseño”, se tratan todos los temas acerca de cómo se ha definido y diseñado la aplicación, los requisitos del sistema, los casos de uso, la arquitectura y los diagramas del proyecto.

La siguiente sección trata del desarrollo de la aplicación, los diferentes componentes que forman el sistema, cómo están diseñados y cómo se comunican entre sí. Cómo están formadas la partes back-end y front-end, las bases de datos y qué tecnologías se han empleado en el proyecto.

Por otra parte, en la sección de “Integración y pruebas”, se comentan las distintas pruebas que se han realizado para verificar el correcto funcionamiento del sistema completo.

En último lugar, se encuentra el apartado de “Conclusiones y trabajo futuro”, en el cual se realiza un análisis del trabajo realizado y el que se llevará a cabo en futuras versiones de la aplicación.

2 Estado del arte

2.1 Sector de la salud

La sanidad privada en España es un sector que a pesar de la crisis no ha parado de crecer en los últimos años. Desde el inicio de la crisis en 2008, el crecimiento de la facturación ha aumentado casi un 4% anual [3], y con una previsión de continuación de esta tendencia durante los próximos años. Este crecimiento se ha debido también al descontento con la sanidad pública desde el inicio de la crisis, por los recortes, el aumento de las listad de espera, la externalización de servicios y la introducción del copago sanitario.

Sólo en el sector privado, el sector de la salud mueve anualmente más de 30.000 millones de euros, mientras que en el sector público se estima un movimiento anual de 70.000 millones de euros, lo que supone cerca del 2,7% y 6,5% del PIB respectivamente [4][5]. En el año 2018, el gasto en sanidad pública creció un 3,9% con respecto al año 2017, un total de 4.253 millones de euros [6][7][8], con un crecimiento del 2,6% entre 2013 y 2017 [9].

Todos estos datos, junto al atraso tecnológico del sector con respecto a otros, indican una continuación al alza en su expansión y evolución durante los próximos años, en los cuales será necesaria una modernización y adaptación de las tecnologías actuales a las nuevas tecnologías.

2.2 Aseguradoras médicas

El sector de la sanidad privada no ha parado de crecer durante los últimos años, superando los 11 millones de asegurados en España, aproximadamente el 21% de la población [10][11], con un crecimiento constante cercano al 4% anual desde 2015 [12].

Existen un elevado número de aseguradoras privadas, entre las principales se encuentran Adeslas, con cerca de los dos millones y medio de asegurados, una cuota de mercado del 32,83%, más del doble que la segunda, Sanitas, con aproximadamente un millón de asegurados, un 16,00% de la cuota de mercado, seguidas de Asisa (15,52%), DKV Seguros (5,97%) y Mapfre España (4,88%) [5].

Actualmente tan sólo un número reducido de aseguradoras disponen de un sistema de citación online, pero ninguna dispone de un sistema de citación que les permita conectar con todos los sistemas de agendas disponibles en el mercado.

Gracias al Hub de Citación Online, les evitamos a las aseguradoras el proceso de implementación de la conexión con cada agenda médica, actuando como intermediarios, y reduciendo la carga de trabajo a una única conexión con nuestro Hub.

2.3 Agendas médicas

Existen numerosas agendas médicas, usadas por las diferentes clínicas y hospitales, para que los facultativos puedan organizar las citas de sus pacientes y facturar a sus clientes a través de las compañías aseguradoras.

Entre las principales agendas médicas disponibles en el mercado, se encuentran TVS+ [13], Axon, Stacks, Infomed, Info33, Q-Soft, Ekon y tuOtempO.

Para que los diferentes softwares de agendas disponibles puedan ser integrados en el Hub de Citación Online, deben disponer de una API de servicios web, para que se puedan consumir de forma online para solicitar los servicios, realizando una conexión por cada una de las agendas.

2.4 Servicios médicos online y citación médica

En la actualidad, en el sector sanitario está atrasado tecnológicamente con respecto a otros sectores como por ejemplo la banca o las telecomunicaciones, a pesar de los avances de los últimos años.

La razón de este atraso se debe principalmente a la existencia de múltiples aseguradoras y agendas médicas y a la falta de acuerdos de unificación de estándares. Cada sistema está estructurado y desarrollado de maneras distintas, y aunque existen algunos estándares, como por ejemplo HL7, DICOM o IHE, no se siguen de manera exhaustiva.

Hasta el momento, debido a la existencia de múltiples aplicaciones de agendas médicas, cada aseguradora que quiera implementar un sistema de citación online necesita realizar una integración con cada agenda disponible, siendo este un proceso complejo y largo de realizar.

Gracias al Hub de citación online, la complejidad de las conexiones con cada agenda queda reducida a una única integración con el sistema de HCO. Quedando solucionado el problema de la falta de estándares y cohesión entre las agendas.

3 Análisis y diseño

3.1 Descripción del sistema

En esta sección se realiza una descripción más exhaustiva del Hub de Citación Online, cómo está definido, los requisitos necesarios, la arquitectura y cuáles son los usuarios que interactuarán con la aplicación.

3.1.1 Descripción y motivación

Esta aplicación es una solución para que las aseguradoras médicas puedan conectarse con las diferentes agendas médicas disponibles en el mercado, sin necesidad de realizar una integración individual con cada una de ellas.

La complejidad del proceso de conexión con cada agenda médica la soporta el Hub de Citación Online, siendo este proceso una caja negra para las aseguradoras, las cuales sólo necesitan realizar una conexión con el propio sistema a través de los servicios ofrecidos por su API REST.

Cada aseguradora puede desarrollar su propio front-end desde el cual realizar peticiones a la API y consumir sus servicios. Además, para las aseguradoras médicas que deseen abstraerse casi por completo del proceso de conexión, se ofrece el servicio del denominado "Modo delegado".

El Modo delegado consiste en un front-end, con estilos personalizados para cada compañía, que consume los servicios de la API. La única complejidad que supone para las aseguradoras es realizar un cifrado de los datos de sus asegurados y realizar una redirección a la URL proporcionada. A partir de ese momento, se delega la responsabilidad y el control al Modo delegado, desde donde el asegurado puede solicitar su cita, anular las ya solicitadas y consultar su historial de citas pedidas.

3.1.2 Usuarios

El sistema interactúa con cuatro tipos diferentes de usuarios:

- **Asegurados:** a través de sus respectivas aseguradoras, se identificarán con sus datos y solicitarán citas, las anularán o consultarán su historial. Estas acciones se podrán realizar a través de nuestro "Modo delegado" o a través de la parte frontal que creen las aseguradoras, siendo estas las que manejarán los datos de sus asegurados.

- **Aseguradoras:** se encargan de enviar los datos de los asegurados al sistema. Existen dos opciones para esto. La primera es realizar directamente peticiones a la API, encargándose de mostrar los datos devueltos a sus asegurados a través de un frontal propio. La segunda opción es delegar la responsabilidad de realizar las peticiones en el "Modo delegado". Esta es la parte frontal que se ofrece a las aseguradoras para agilizar el proceso de conexión, siendo necesaria únicamente una redirección con los datos del asegurado encriptados.

- **Agendas médicas:** son las encargadas de ofrecer sus servicios online para que puedan ser consultados mediante peticiones. Tienen que disponer de APIs de servicios web para que puedan ser consumidos a través del Hub, y así poder solicitar citas, anularlas y consultar a las diferentes opciones que ofrecen.

- **Modo delegado:** se trata de la parte frontal, desarrollada con el framework de aplicaciones web Angular, a la que se accede mediante una redirección a través de la web de cada aseguradora. Está diseñada para reducir la carga de trabajo a las aseguradoras para que la integración sea más sencilla. Se encarga de realizar peticiones a la API del Hub y mostrar a los asegurados los datos y opciones de la cita y su historial.

3.2 Definición del sistema

3.2.1 Casos de uso

A continuación, se definen los casos de uso principales del sistema de servicios ofrecidos por la API.

Identificador: HCO-001	**Nombre Caso de Uso:** Solicitud de horarios de una agenda.
Actores involucrados: Aseguradoras que realicen la petición con los datos de los asegurados. HUB que traduce los datos solicitados y realiza las peticiones a las APIs de las agendas. Agendas médicas que devuelven los datos a través de su API.	
Resumen: La aseguradora realizará una petición al HUB de Citación Online con los datos de su asegurado y se le devolverán los huecos disponibles de los horarios de la agenda médica correspondiente.	
Precondiciones:	

El asegurado ha entrado en la web de su aseguradora y ha seleccionado los datos de la cita.	
Postcondiciones: Se devuelve una lista de horarios disponibles para la agenda correspondiente a partir de los datos introducidos por el asegurado.	
Curso Básico de Eventos	
Usuario	**Sistema**
1. La aseguradora realiza una petición con los datos seleccionados por el asegurado.	2. El sistema consulta la agenda correspondiente a partir de los datos recibidos en la petición. Devuelve una respuesta con la lista de horarios de la agenda correspondiente con los huecos disponibles.
Caminos Alternativos	
2a. El sistema no consigue encontrar la agenda. 2a.1. Devuelve una respuesta notificando que no existe ninguna agenda médica con los datos seleccionados. 2b. El sistema consulta una agenda que no tiene horarios definidos. 2b.1. Devuelve una respuesta notificando que no existen horarios disponibles para esa agenda.	

Identificador: HCO-002	**Nombre Caso de Uso:** Consulta de historial de citas de asegurado.
Actores involucrados: Asegurados que deseen consultar su historial de citas solicitadas y anuladas. Aseguradoras que realicen la petición con los datos de los asegurados. HUB que traduce los datos solicitados y realiza la consultad en la base de datos.	
Resumen: Una vez que un usuario ha solicitado alguna cita en una agenda médica, puede consultar su historial de citas a través de su aseguradora. La aseguradora realiza una petición al HUB de Citación Online, el sistema consultará a las agendas correspondientes y devolverá una lista con las citas que ha realizado ese usuario.	
Precondiciones: El asegurado ha entrado en la web de su aseguradora y ha solicitado consultar su historial.	
Postcondiciones:	

Se muestra una lista de citas que han sido solicitadas por ese asegurado.	
Curso Básico de Eventos	
Usuario	**Sistema**
1. El usuario solicita su historial de citas a través de su aseguradora. La aseguradora realiza una petición con la solicitud al HUB de Citación Online.	2. El sistema consulta a las agendas correspondientes y devuelve las citas del asegurado guardadas en la BBDD que se han solicitado a través del HUB de citación online.
Caminos Alternativos	
2a. El sistema consulta las agendas correspondientes y no obtiene ninguna cita para ese asegurado. 2a.1. El sistema devuelve un mensaje notificando que ese asegurado no tiene ninguna cita guardada en su historial.	

Identificador: HCO-003	**Nombre Caso de Uso:** Solicitud de cita de asegurado.
Actores involucrados: Aseguradoras que realicen la petición con los datos de los asegurados. HUB que traduce los datos solicitados y realiza las peticiones a las APIs de las agendas. Agendas médicas que devuelven los datos a través de su API.	
Resumen: La aseguradora realiza una petición al HUB de Citación Online con los datos seleccionados por sus asegurados, que la solicitará a la agenda correspondiente.	
Precondiciones: El usuario ha solicitado la lista de horarios para una agenda a través de su aseguradora. (Consultar Caso de Uso *THORARIO*).	
Postcondiciones: Se obtiene una confirmación de solicitud de cita con los datos y la fecha seleccionados.	
Curso Básico de Eventos	
Usuario	**Sistema**
1. La aseguradora realiza una petición al HUB de Citación Online con los datos seleccionados por su asegurado.	2. El sistema obtiene los datos. Reserva el hueco para el asegurado solicitante y devuelve una respuesta con la confirmación de la cita a la aseguradora que ha realizado la

	petición.
Caminos Alternativos	
2a. El hueco solicitado ya no está disponible. 2a.1. El sistema devuelve una respuesta indicando que el hueco seleccionado ya ha sido reservado.	

Identificador: HCO-004	**Nombre Caso de Uso:** Anular cita de asegurado.
Actores involucrados: Aseguradoras que realicen la petición con los datos de los asegurados. HUB que traduce los datos solicitados y realiza las peticiones a las APIs de las agendas. Agendas médicas que devuelven los datos a través de su API.	
Resumen: Una vez que un usuario ha pedido una cita de una agenda médica, puede realizar una solicitud de anulación para esa cita a través de su aseguradora. La aseguradora realiza una petición al HUB de Citación Online, y a continuación, el sistema se encarga de anular la cita en la agenda médica correspondiente.	
Precondiciones: El asegurado ha entrado en la web de su aseguradora y ha consultado sus citas a través de su historial.	
Postcondiciones: El asegurado recibe una confirmación de la anulación de cita a través de su aseguradora.	
Curso Básico de Eventos	
Usuario	**Sistema**
1. El usuario anula la cita solicitada a través de su aseguradora. La aseguradora realiza una petición al HUB de Citación Online con la solicitud de anulación de la cita seleccionada.	2. El sistema recibe la solicitud de anulación y realiza la petición de anulación a la agenda correspondiente. En caso de que la anulación se haya realizado correctamente devuelve una confirmación al asegurado.
Caminos Alternativos	
2a. El sistema no puede anular la cita, ya sea porque la fecha de la cita es muy cercana o por otros motivos. 2a.1. El sistema devuelve una respuesta notificando que la cita no ha podido ser anulada.	

Identificador: HCO-005	**Nombre Caso de Uso:** Solicitud de lista de agendas
Actores involucrados: Aseguradoras que realicen la petición al HUB con los datos. HUB que traduce los datos solicitados y realiza las peticiones a las APIs de las agendas. Agendas médicas que devuelven los datos a través de su API.	
Resumen: Para consultar el listado de agendas, ya sean para una aseguradora o para un centro y especialidad, se realiza una petición al sistema. La respuesta a esta petición es una lista con las agendas disponibles respecto a los datos solicitados.	
Precondiciones: Un asegurado a entrado en la web de su aseguradora para consultar los horarios disponibles y ha introducido los datos de filtrado.	
Postcondiciones: El sistema devuelve un listado con las agendas disponibles para los datos seleccionados por el asegurado a través de su aseguradora.	
Curso Básico de Eventos	
Usuario	**Sistema**
1. La aseguradora, a partir de los datos de filtrado introducidos por su asegurado, realiza una petición al sistema para consultar las agendas disponibles para que sean seleccionadas.	2. El sistema consulta a la base de datos a partir de los datos recibidos, y devuelve un listado de agendas.
Caminos Alternativos	
1a. La aseguradora solicita el listado con unos datos de filtrado no almacenados en la base de datos. 2a. El sistema devuelve un listado de agendas vacío.	

3.2.2 Requisitos del sistema

3.2.2.1 Requisitos funcionales

- **RF1:** el usuario debe ser capaz de solicitar una cita a través de la web de su aseguradora médica.
- **RF2:** el usuario debe ser capaz de visualizar los horarios disponibles para una agenda a través de la web de su aseguradora médica.
- **RF3:** el usuario debe ser capaz de consultar su historial de citas solicitadas de forma online a través de la web de su aseguradora médica.
- **RF4:** el usuario debe ser capaz de anular una cita a través de la web de su aseguradora médica.
- **RF5:** el usuario debe ser capaz de elegir entre las diferentes agendas disponibles para una especialidad en un mismo centro médico.
- **RF6:** la aplicación debe permitir a las aseguradoras consumir los servicios RF1, RF2, RF3, RF4, RF5 a través de una API.
- **RF7:** las aseguradoras médicas deben disponer de un front-end denominado "Modo delegado" para integrarse con el sistema sin necesidad de integrarse con la API.
- **RF8:** el usuario debe ser capaz de consumir los servicios web RF1, RF2, RF3, RF4 y RF5 a través del "Modo delegado".
- **RF9:** el "Modo delegado" debe ser multilenguaje, con capacidad de cambiar de idioma desde dentro de la aplicación.
- **RF10:** el sistema debe mostrar un "look and feel" diferente según la aseguradora que realice la petición.
- **RF11:** el sistema debe disponer de un sistema de OTP (one-time password) para que los asegurados verifiquen con su teléfono o su email su solicitud.

3.2.2.2 Requisitos no funcionales

- **RNF1:** el sistema de la API debe estar codificado en Java EE 8 [14].
- **RNF2:** la parte front-end del sistema debe estar desarrollado con el framework Angular [15].
- **RNF3:** el "Modo delegado" debe ser una aplicación web responsive, soportada tanto en modo escritorio como en móvil.
- **RNF4:** el sistema debe estar desplegado en la nube de RedHat OpenShift [16].

- **RNF5:** el sistema debe hacer uso de las bases de datos en la nube de Amazon Web Services [17].

- **RNF6:** el sistema debe disponer de logs que reflejen las peticiones y acciones que se producen cuando se realiza una petición.

- **RNF7:** la arquitectura del sistema debe permitir trasladar con facilidad la aplicación entre las diferentes nubes disponibles.

- **RNF8:** se debe usar Git como software de control de versiones [18].

3.3 Arquitectura del sistema

El sistema actual está formado por una arquitectura orientada a microservicios separada en tres capas, la capa de presentación, la capa de lógica de negocio y la capa de acceso a datos.

El sistema tiene un componente principal, que es la API de servicios web Java EE [14] a la que realizan peticiones las compañías aseguradoras, definidos en el diagrama como "HCO". Gracias a estos servicios se pueden consultar las agendas, solicitar y anular citas en las agendas disponibles y consultar el historial de citas solicitadas.

Para ofrecer estos servicios, desde este back-end se implementan las llamadas a las diferentes agendas, algunas ofrecidas como servicios SOAP y otras como servicios REST.

Además de los servicios principales, el sistema dispone de un componente secundario, denominado "Modo Delegado", el cual se ofrece para actuar como frontal a las compañías aseguradoras que lo deseen, en lugar de implementar ellas uno propio.

El "Modo delegado" tiene dos componentes, un front-end, desarrollado con el framework Angular, que realiza peticiones a una parte back-end desarrollada en Java EE, la cual a su vez consume los servicios realizando peticiones a la API principal.

La parte back-end del "Modo delegado" se ha creado para añadir servicios exclusivos para el front-end separados de los servicios ofrecidos a las aseguradoras.

Ambos back-ends, tanto el de los servicios principales de HCO como el de "Modo delegado" tienen acceso a una base de datos relacional Amazon Aurora [19], la cual es compatible con MySQL y PostgreSQL, donde se almacenan los datos de las agendas, los centros, las citas...

El "Modo delegado" hace uso de un servicio notificaciones, el cual se envía SMS o emails a los asegurados con un OTP (one-time password), un token utilizado para evitar ataques por replicación, que es introducido en el momento en que se quiere consumir un servicio a través del front-end cuando el asegurado no está identificado.

Por último, se almacenan los datos de auditoría de las peticiones que se han realizado al sistema a través de una base de datos no relacional Amazon DynamoDB [20].

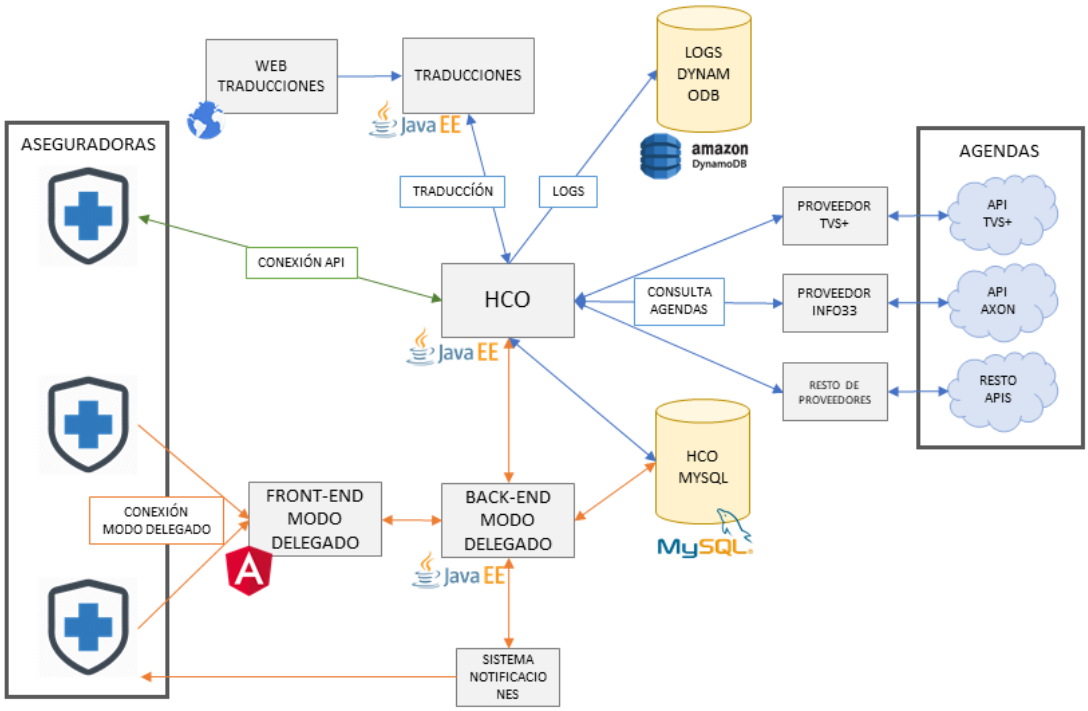

Ilustración 2: Diagrama de arquitectura

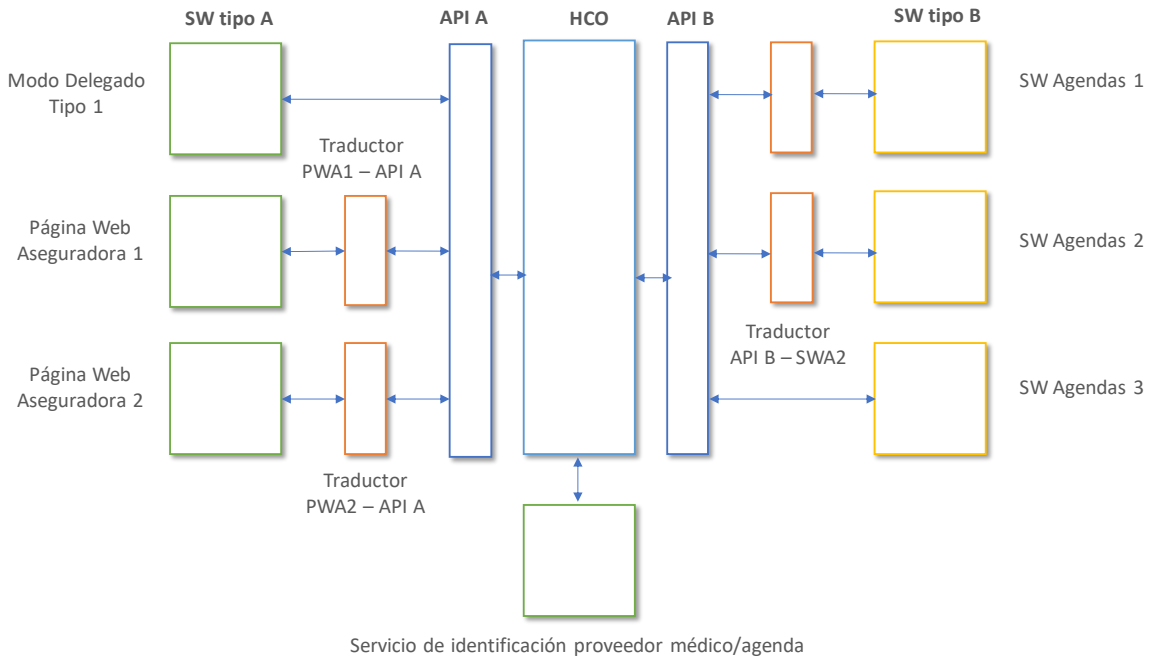

Ilustración 3: Diagrama de alto nivel de arquitectura

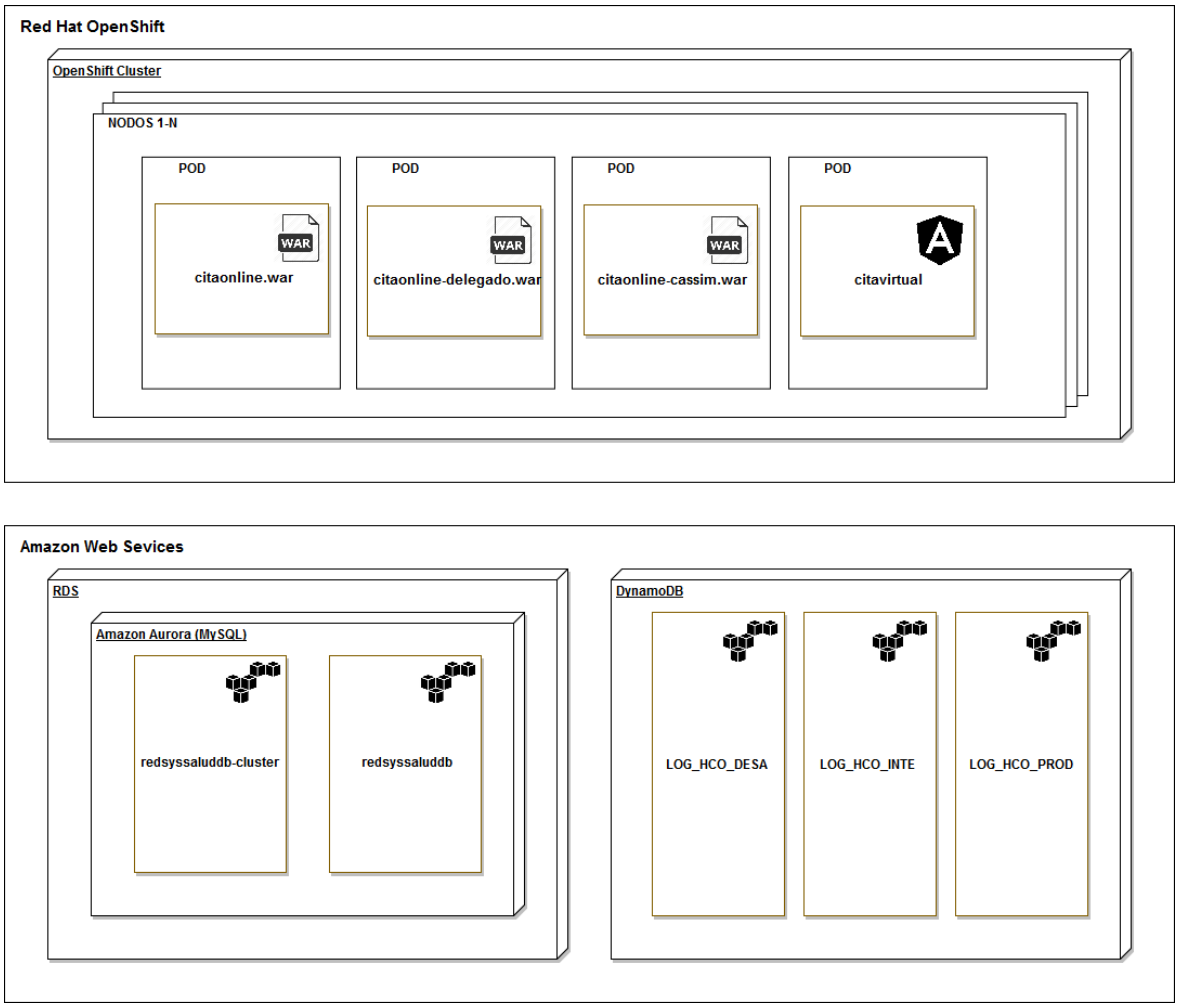

Ilustración 4: Diagrama de despliegue

3.4 Diagramas

3.4.1 Diagramas de secuencia

Los diagramas de secuencia se utilizan para modelar la interacción entre los diferentes objetos y actores de un sistema a través del tiempo para cada caso de uso.

A continuación, se muestran los dos diagramas de secuencia más relevantes del proyecto, el de solicitar cita y el de consultar historial de citas.

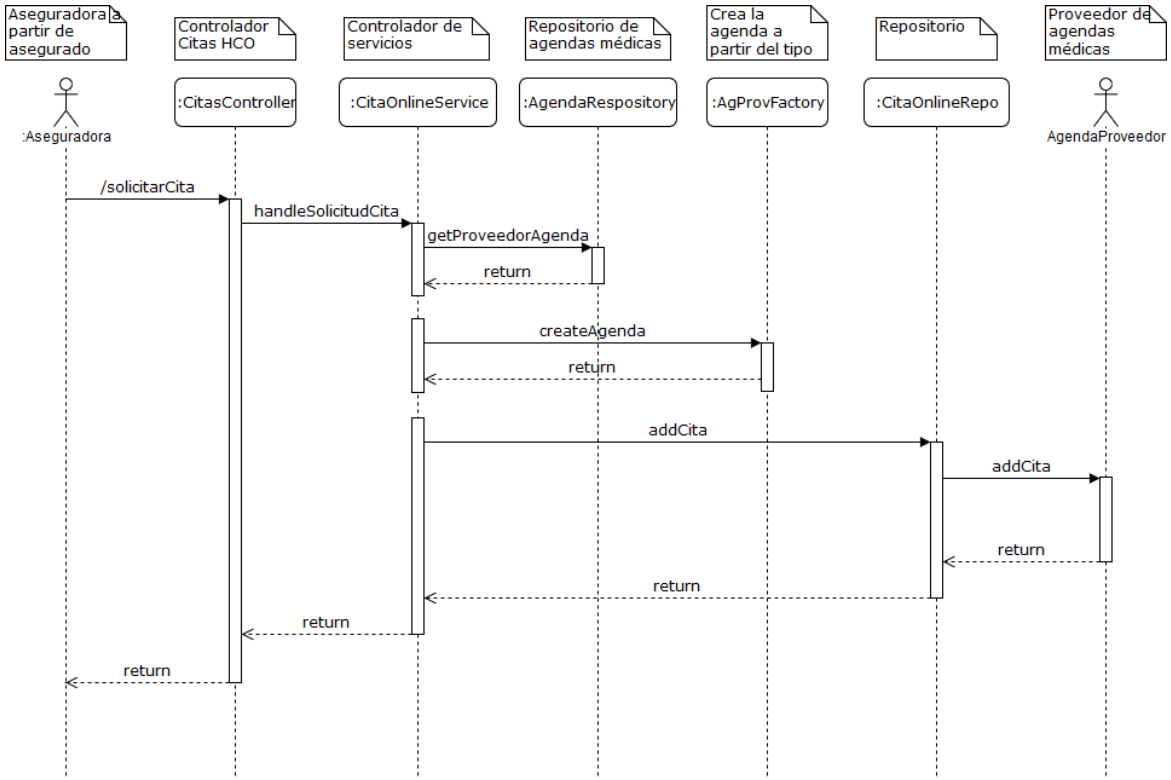

Ilustración 5: Diagrama de secuencia de "Solicitud de cita"

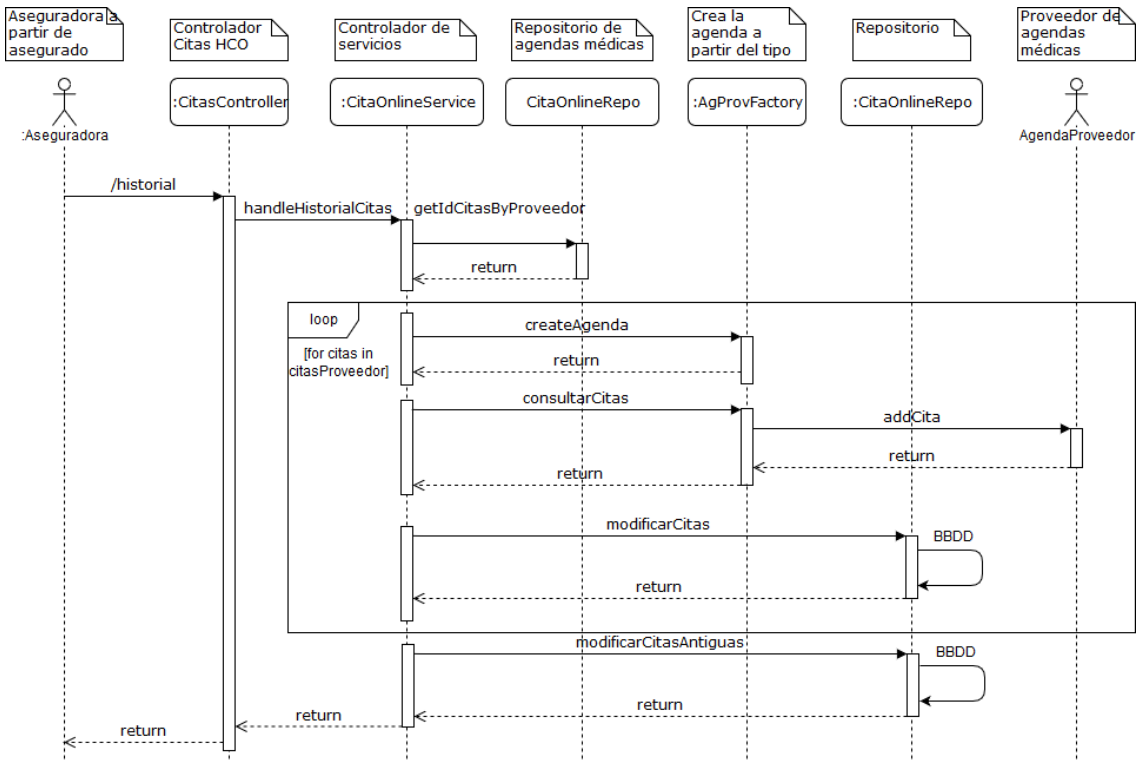

Ilustración 6: Diagrama de secuencia de "Consulta de historial"

3.4.2 Diagramas de flujo

Los diagramas de flujo describen gráficamente el recorrido y los diferentes pasos de un proceso. Los diagramas que se están representados en este documento representan las funcionalidades implementadas en el sistema.

En este caso, se han representado únicamente las funcionalidades básicas de solicitar cita y la de consultar historial.

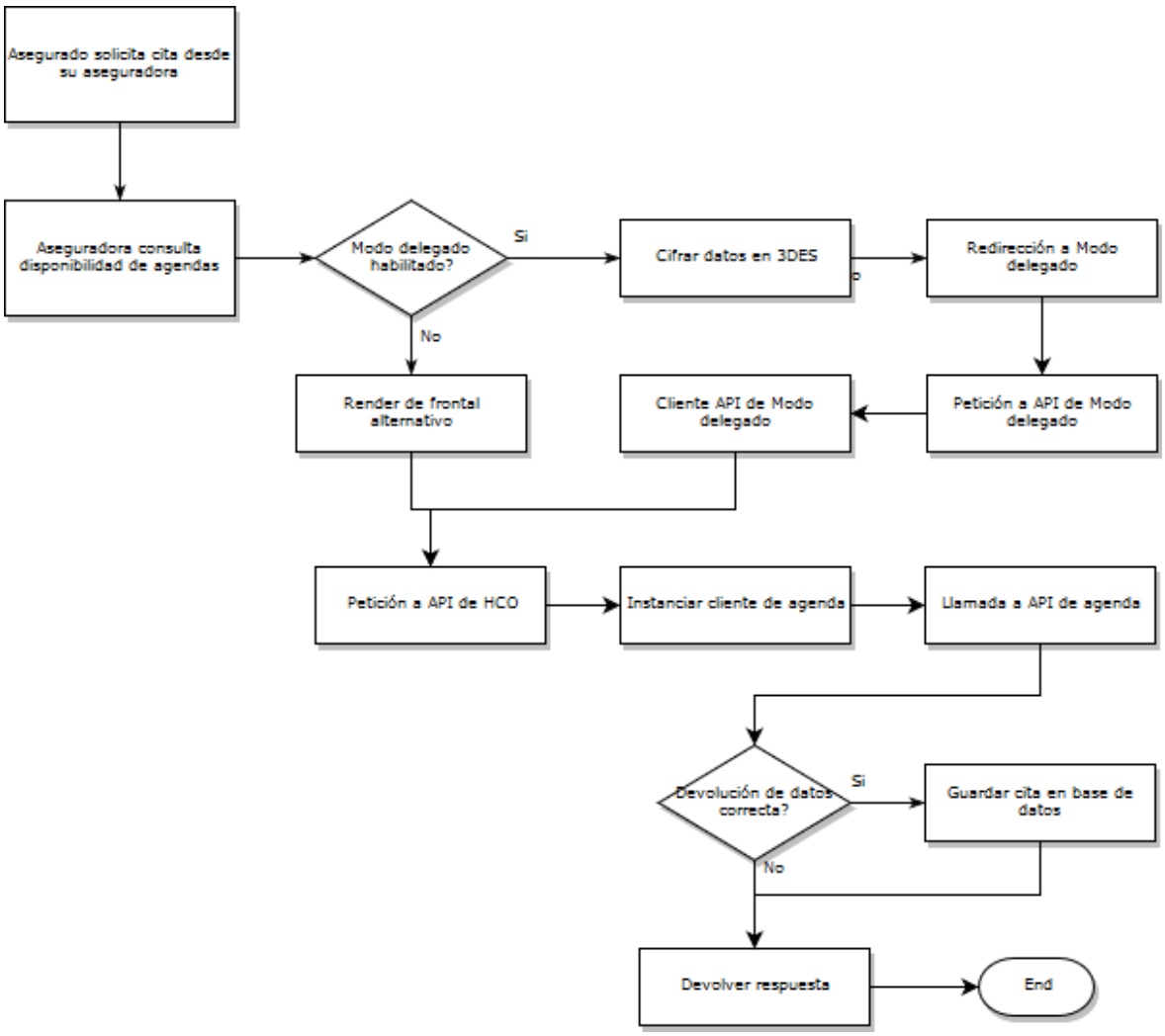

Ilustración 7: Diagrama de flujo de "Solicitud de cita"

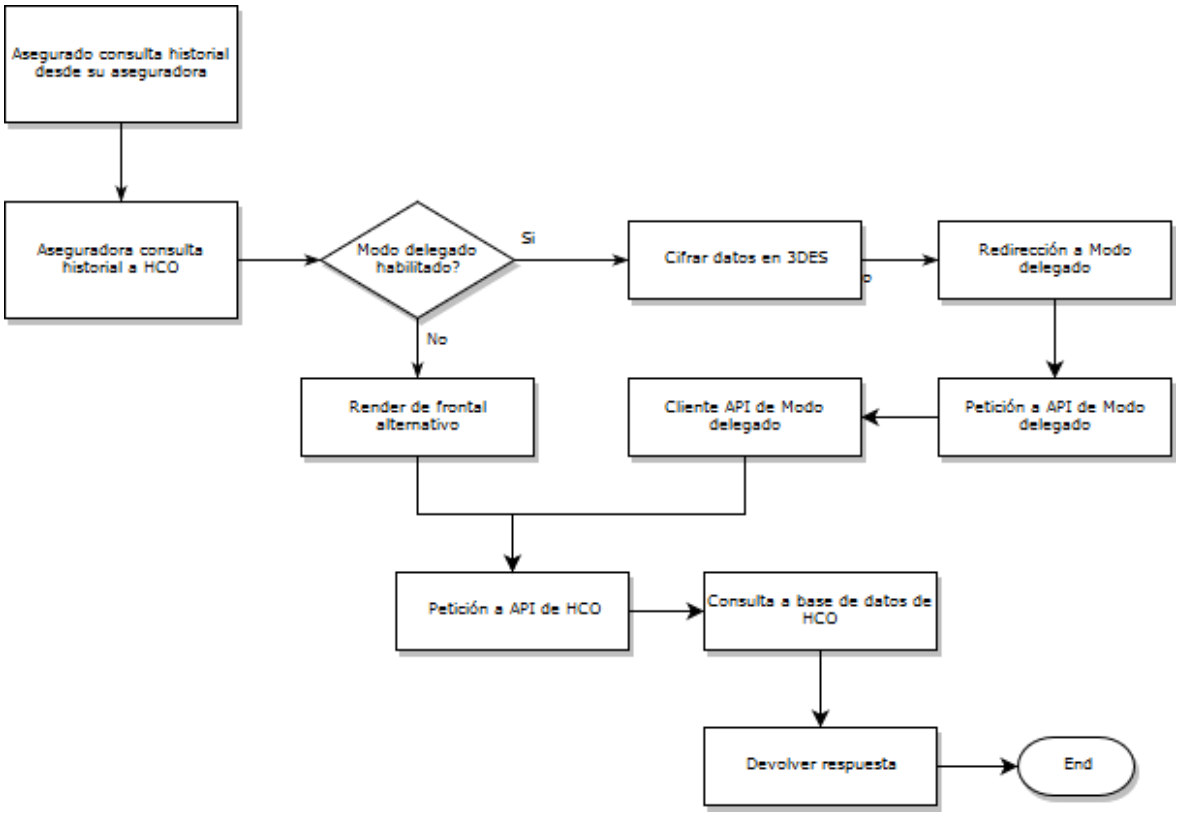

Ilustración 8: Diagrama de flujo de "Consulta de historial"

4 Desarrollo

4.1 Back-end

La parte back-end de una aplicación se abarca todos los componentes que se encuentran en el lado del servidor, encargándose de la manipulación de datos y sirviendo a la parte del front-end.

Esta aplicación está desarrollada con el framework open-source Spring [21], sobre la plataforma Java EE [14]. Hace uso de la herramienta Maven [22], para la contrucción de proyecto y la gestión de dependencias.

4.1.1 Java EE

Java EE o Java Enterprise Edition [14] es una plataforma de programación destinada al entorno empresarial, que extiende las funcionalidades de Java SE (Standard Edition). Añade especificaciones tales como servicios web, computación en la nube, JDBC para la conexión con bases de datos, entre muchas otras.

La última versión, Java EE 8, fue liberada en agosto de 2017, pero dispone de un largo recorrido, siendo su primera versión de finales de 1999. A finales de 2017, Oracle, la compañía propietaria del lenguaje Java y Java EE, cedió los derechos de Java EE a la compañía Eclipse Foundation, donde cambió el nombre a Jakarta EE,

Se ha elegido la construcción de la parte back-end del proyecto en Java, además de por ser uno de los lenguajes más populares desde hace varios años, por ser el lenguaje que se utiliza mayoritariamente en Redsys España en casi todos los proyectos.

En el momento del desarrollo de esta aplicación, se necesitó disponer de un entorno ágil de trabajo, que permitiera cambios y despliegues continuos. Una vez que la aplicación llegue a una fase estable, se subirá a los entornos de Redsys España para que sea controlada y monitoreada por un departamento encargado precisamente de esa tarea.

4.1.2 Apache Maven

Lanzado inicialmente en 2004, Maven [22] es una herramienta dedicada a automatizar la construcción de proyectos: gestiona sus dependencias (las librerías que son necesarias para la ejecución de la aplicación), compila, empaqueta y pasa las pruebas de ejecución.

Maven simplifica enormemente estas tareas, siendo necesario únicamente disponer de un fichero llamado “pom.xml”, donde se definen todos los datos que necesita el proyecto. Una vez creado el fichero, con comandos simples se descargan las dependencias, se compila, se despliega y se pasan las pruebas.

4.1.3 Spring Framework

Spring es un framework de código abierto para el desarrollo de aplicaciones Java, lanzado en el año 2002. Es un framework que no impone ningún modelo de programación en particular, pero dispone de múltiples arquitecturas, como Spring Boot, Spring Roo o Spring Cloud, entre muchos otros.

En nuestro caso en particular utilizamos Spring MVC configuration annotation de Spring Framework, para realizar la parte del back end, la API REST del Hub de Citación Online. Permite prescindir de ficheros XML, utilizando únicamente anotaciones. Esto posibilita la visualización de los errores en tiempo de compilación, evitando numerosos problemas.

Otra de las principales ventajas y razones del uso de Spring Framework, es que reduce notablemente las líneas de código, puesto que autogenera parte de ellas internamente, reduciendo el tiempo de desarrollo de la aplicación. Además, dispone de múltiples librerías y módulos que facilitan el acceso a bases de datos y en general ayudan a desarrollar el proyecto.

4.1.4 Traductor

A pesar de no haber trabajado directamente en el módulo de traducción, es importante nombrarlo por ser una pieza tan importante del proyecto.

El sistema de traducción es una pieza fundamental de la aplicación. Permite trabajar con los códigos internos de cada aseguradora y cada agenda de manera agnóstica, realizando las traducciones de códigos pertinentes de cada elemento. Las aseguradoras envían las peticiones al Hub con sus códigos, cuando llegan, se traducen al lenguaje universal que se utiliza internamente en la aplicación y una vez se necesita realizar una petición a la agenda pertinente, se traduce el lenguaje universal a los códigos de la agenda.

Se trata de una API REST, codificada a partir de un generador de código llamado Telosys [23], disponible para línea de comandos y como plugin para Eclipse [24]. Se realizan peticiones desde HCO a la API con la finalidad de que esta realice las traducciones necesarias para la aplicación.

Permite realizar traducciones de los elementos que sean necesarios, como identificadores de centros, de especialidades, de realizadores o de aseguradoras. Es configurable a bajo nivel, permitiendo realizar únicamente las traducciones de los campos necesarios para cada petición.

Como ejemplo, supongamos que una aseguradora realiza una petición de solicitud de cita al Hub de Citación Online, pasando su identificador interno de centro y de especialidad. A continuación, HCO realiza una petición a la API del sistema de traducciones, para obtener los códigos internos de la agenda médica a la que se va a solicitar la cita. Una vez obtenida la respuesta de la API, se procede a realizar la petición correspondiente al software de agendas necesario para reservar la cita.

4.2 Front-end

La parte front-end de una aplicación se abarca todos los componentes que se encuentran en el lado del cliente, interactuando con los usuarios. Se encarga de recopilar los datos de entrada del usuario, transmitiéndoselos a la parte back-end en el formato que esta precisa, y mostrando su respuesta.

Esta aplicación está desarrollada con Angular [15], y codificada en TypeScript [25], HTML y con el metalenguaje de CSS, SASS [26], que extiende las capacidades y funcionalidades de este.

La sección de front-end será la más extensa, por ser en la que más tiempo he estado implicado y más responsabilidad he tenido.

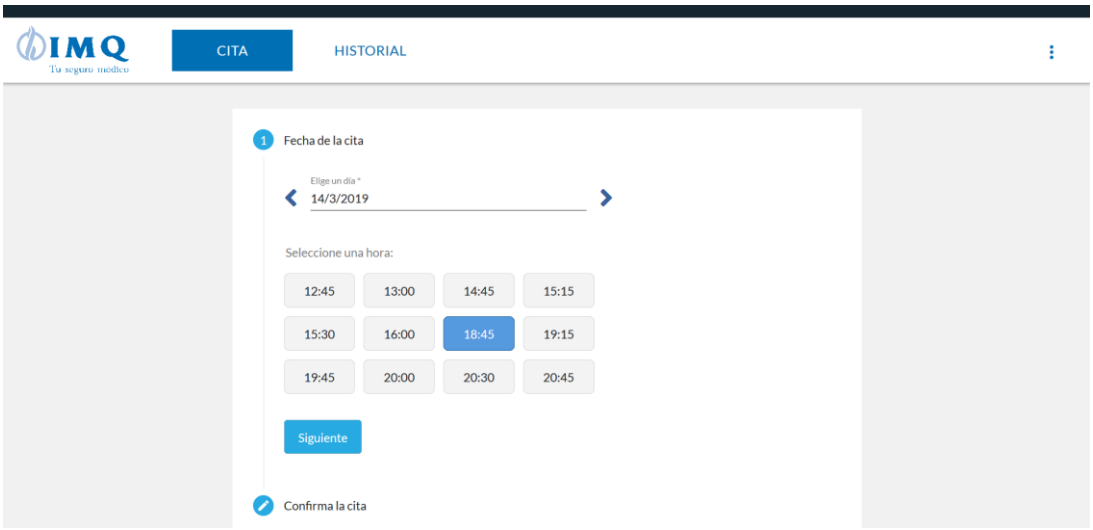

Ilustración 9: Captura de "Modo delegado" en vista de escritorio

4.2.1 Angular

Angular se trata de un framework open-source, desarrollado por Google, para crear aplicaciones web SPA (Single-page Application) de una sola página [27]. Facilita el desarrollo y las pruebas, utilizando la arquitectura MVC (Modelo Vista Controlador) en la creación de aplicaciones.

El "Modo delegado" es la aplicación front-end del Hub de Citación Online que está creada con el framework de Angular. Se creó inicialmente con Angular 6, y se actualizó cuando se lanzó la versión 7. Se pretende mantener actualizada la aplicación según vayan saliendo actualizaciones de Angular, las cuales se liberan cada 6 meses.

Angular tiene el beneficio de poder instalar paquetes y librerías externas gracias a "npm", el manejador de paquetes por defecto para Node.js [28].

Para acceder al “Modo delegado”, es necesario que a través de las compañías aseguradoras se genere un enlace, pasando por parámetros los datos encriptados del cliente y la compañía. A través de este proceso, las compañías aseguradoras “delegan” el control al frontal de Angular, accediendo a la aplicación a través del enlace generado.

Desde la aplicación front-end se realizan las peticiones necesarias al back-end para obtener los datos, realizar las citaciones o consultar el historial de citas.

En primer lugar, al acceder a la aplicación se realiza una petición al back-end para desencriptar los datos y poder trabajar con ellos guardándolos en variables de sesión. A partir de ese momento, dependiendo de las acciones que realice el asegurado, se reservarán citas, seleccionando la cita a partir de las fechas y las horas disponibles, consultará su historial de citas o anulará las citas que aún no hayan sido atendidas.

Ilustración 10: Captura de “Modo delegado” en vista de escritorio

4.2.2 Angular Material

Angular Material [29] son un conjunto de componentes para Angular, que utilizan la normativa de diseño Material Design [30], para mejorar el diseño y la experiencia de usuario de las aplicaciones.

Material Design nació con la intención de proporcionar un diseño intuitivo, cómodo y ergonómico para las personas, con colores, fuentes, iconos, estructuras y texturas minimalistas.

Angular Material dispone de numerosos elementos que aligeran el desarrollo. En el front-end de “Modo delegado” utilizamos algunos de los componentes de Angular Material como el menú de navegación, tablas de datos para mostrar el historial, selectores de fechas, formularios para la selección de cita y botones, entre otros.

4.2.3 Bootstrap

Bootstrap [31] es la librería más popular para el desarrollo front-end de aplicaciones web. Provee de elementos para el desarrollo de HTML, CSS y JavaScript, tales como formularios, botones, tipografía o animaciones.

En el caso de nuestra aplicación, complementa la funcionalidad que no se consigue con Angular Material, como por ejemplo la creación de parte de los botones o las estructuras de algunos componentes.

4.2.4 Diseño responsive

La parte frontal del proyecto, "Modo delegado", se distingue por tener un diseño web responsivo (responsive design), también llamado adaptable o adaptativo. Nace a raíz de la existencia de múltiples dispositivos, desde smartphones a tabletas o monitores de escritorio, cada uno con tamaños de pantalla diferentes.

El diseño responsivo es un enfoque del diseño web que se caracteriza por la adaptación de las vistas de una página web al tamaño de la pantalla y características del dispositivo que esté navegando por ellas. Gracias a esto, con un único diseño de la página, su visualización será correcta en todos los dispositivos.

Para lograr un diseño responsivo, se utilizan media queries, un módulo de CSS3 que permite definir los estilos de una página según el tipo de dispositivo, las dimensiones, orientación o modo de la pantalla.

En los anexos de este documento se adjuntan capturas más detalladas de la aplicación.

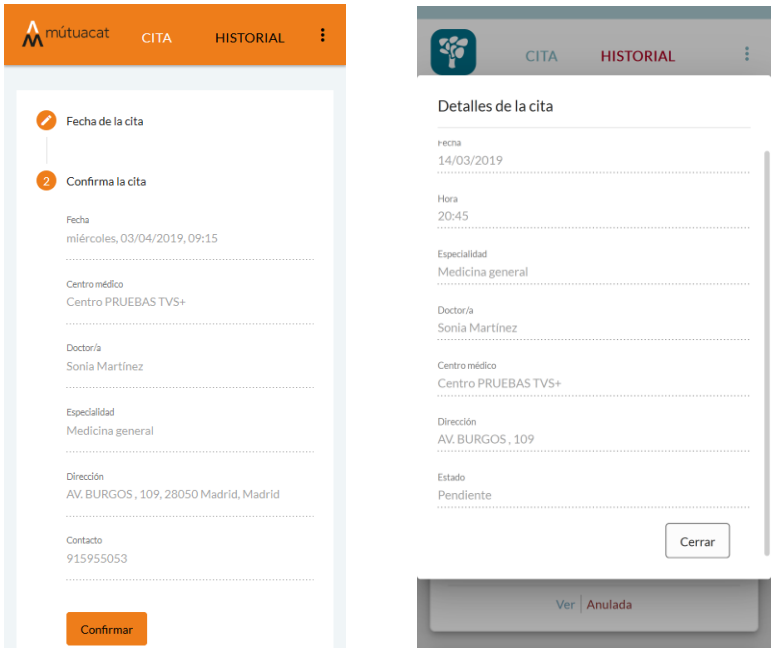

Ilustración 11: Capturas de "Modo delegado" en vista de móvil

4.3 Bases de datos

El sistema de Hub de Citación Online dispone de dos bases de datos para almacenar la información de la aplicación.

Amazon Web Services tiene múltiples servicios en la nube, entre los que se encuentran las bases de datos, de las que hace uso nuestro sistema. Ofrece servicios de bases de datos relacionales, RDS (Relational Database Service), no relacionales como DynamoDB, así como bases de datos de gráficos y bases de datos de series temporales.

Se encuentran replicadas en un cluster de dos servidores, para aumentar la disponibilidad y evitar un punto simple de fallo (SPOF, single point of failure). Esto evita que se pierdan los datos en caso de que una de las máquinas falle, consiguiendo tener una copia de ellos.

4.3.1 Base de datos relacional Amazon Aurora

AWS cuenta con varios motores de base de datos relacionales creados para la nube, por ejemplo, MySQL, PostgreSQL, Oracle, Microsoft SQL Server o la que utilizamos en nuestro caso, Amazon Aurora.

Amazon Aurora es una base de datos relacional en la nube. Tiene compatibilidad con MySQL y PostgreSQL, con su rendimiento y disponibilidad, combinado con la simplicidad de las bases de datos open-source. Además, es varias veces más rápida que MySQL y PostgreSQL, con un coste muy inferior y la seguridad de la infraestructura de Amazon Web Services.

Proporciona además auto escalabilidad, así como la posibilidad de realizar copias de seguridad en la nube a través del servicio de almacenamiento seguro de Amazon S3 [32].

A continuación, se muestra el modelo entidad-relación de la base de datos de HCO, generado con la herramienta MySQL Workbench.

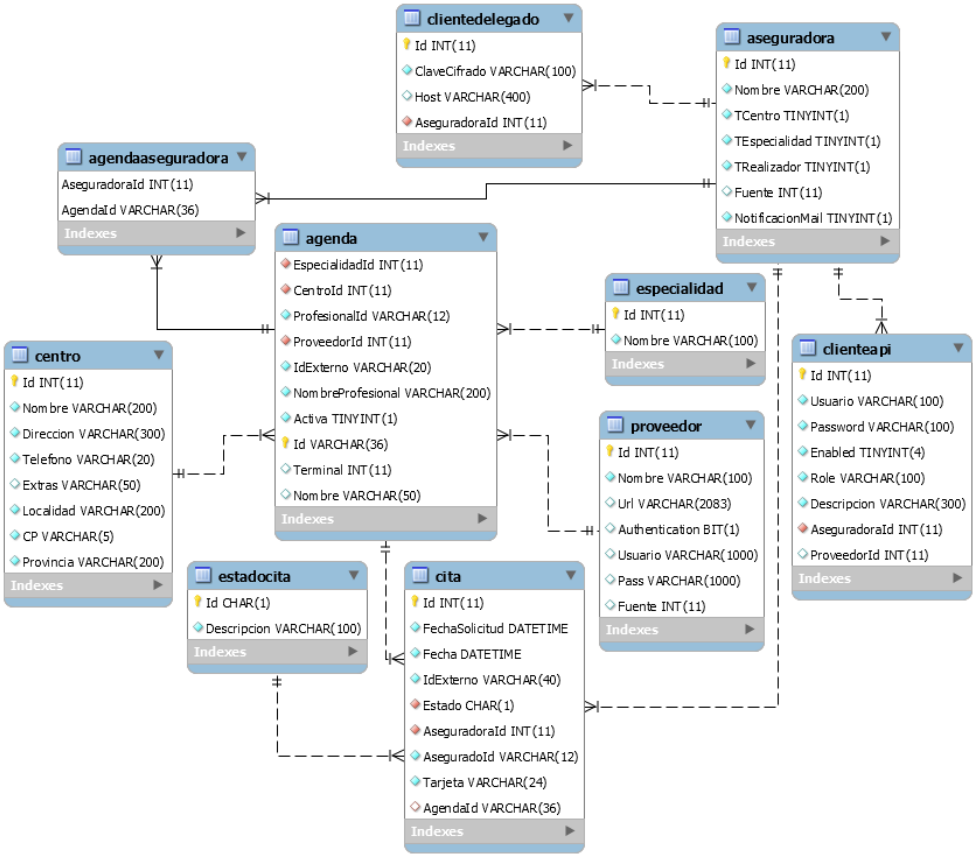

Ilustración 12: Modelo Entidad-Relación

4.3.2 Base de datos no relacional DynamoDB

DynamoDB [20] es el servicio de base de datos no relacional o NoSQL de AWS, que soporta estructuras de tipo clave-valor y documentos.

Al igual que las bases de datos relacionales de las que dispone AWS, DynamoDB cuenta con alta disponibilidad y durabilidad, al contar con replicación a lo largo de múltiples centros de datos. Además, gracias a esto, se disponen de copias de seguridad y puntos de restauración en caso de necesitarlos.

En nuestra aplicación, la base de datos de DynamoDB se utiliza para almacenar todas las peticiones y respuestas que han llegado y se han enviado desde la API.

4.4 AWS (Amazon Web Services)

Aunque se han nombrado previamente algunos de los servicios de Amazon Web Services (AWS) [17] utilizados en el proyecto de HCO, es conveniente dedicar un apartado exclusivamente a esta plataforma.

Amazon Web Services es una filial de la compañía Amazon, considerada una de las plataformas de computación en la nube más importantes, ya actualmente la que más cuota

de mercado abarca, un 33% en 2018. Provee de múltiples servicios de computación en la nube a empresas, gobiernos y personas individuales.

Tiene una filosofía de cobros por servicio utilizado, evitando así pagar por algo de lo que no se hace uso. Por ejemplo, en caso de tener una máquina virtual en la nube, sólo se realiza el cobro la máquina cuando esa máquina está en funcionamiento.

AWS dispone de servicios de computación, almacenamiento, bases de datos, machine learning, blockchain, robótica, IoT (internet de las cosas), entre muchos otros.

El siguiente punto que se va a tratar es uno de los servicios principales de computación en la nube que se utilizan en el proyecto, debido a que los servicios de bases de datos que ofrece AWS ya han sido tratados en apartados anteriores.

4.4.1 Elastic Compute Cloud (EC2)

Amazon EC2 [33] es el servicio principal de computación en la nube de AWS, mediante el cual se permite a los usuarios levantar máquinas virtuales, con varios tipos de imágenes, como Ubuntu, Red Hat Linux, SUSE Linux o Windows Server.

Estas máquinas pueden configurarse para adaptarse a las necesidades del cliente, pudiendo aumentar características como el número de CPUs, la memoria RAM, la capacidad de almacenamiento o el ancho de banda de la red.

Para conectarse a las máquinas virtuales de EC2, es necesario utilizar un protocolo de conexión como SSH, Telnet, X11, RDP (Remote Desktop Protocol, Escritorio Remoto) o VNC (Virtual Network Computing) dependiendo del tipo de instancia de la máquina.

En el caso de HCO, se han utilizado para realizar los despliegues en los entornos de desarrollo, integración y producción, montando un servidor de Apache Tomcat sobre una máquina Linux. En el futuro, para el entorno producción se pretende utilizar el entorno de Red Hat Openshift, debido a que ofrece una capa de abstracción con respecto a las máquinas de EC2 para realizar los despliegues.

4.5 Red Hat OpenShift

OpenShift [34] es un PaaS (Plataform as a Service o plataforma como servicio en castellano) desarrollada por la compañía Red Hat, compuesto por un stack de aplicaciones, la cual permite construir, ejecutar y desplegar aplicaciones contenerizadas con Docker [35] y orquestadas por Kubernetes [36].

Se encarga del mantenimiento de las aplicaciones, automatizando la construcción, los despliegues, la orquestación y escalabilidad de estas. Permite una integración completa con repositorios Git, llegando incluso a automatizar el despliegue de la aplicación cada vez que se hace una subida al repositorio.

Una vez la aplicación se pasa a producción, en lugar del entorno de AWS, se tiene previsto utilizar el entorno proporcionado por OpenShift. Esto es debido a que cuando la aplicación sea estable en el entorno de producción, se migrará al entorno de Redsys España [35], donde se monitorizará para actuar contra incidencias.

4.5.1 Docker

Se trata de un software de código abierto que permite crear imágenes de aplicaciones junto a sus dependencias, entorno y herramientas que necesitan para ser ejecutadas, para ser lanzadas en máquinas virtuales ligeras denominadas contenedores [35].

Esto permite ejecutar una aplicación en cualquier entorno que disponga de Docker, sin necesidad de disponer de una arquitectura específica.

Gracias a la comunidad que existe detrás de la plataforma Docker Hub, se comparten multitud de imágenes para diferentes aplicaciones y entornos preparadas para usarse.

4.5.2 Kubernetes

Kubernetes es un sistema de orquestación de código abierto diseñado para automatizar los despliegues, el escalado y el mantenimiento de aplicaciones contenerizadas [36].

El componente central de Kubernetes es un clúster, compuesto por máquinas físicas o virtuales, que actúan como nodos, en las cuales se levantan uno o más contenedores Docker, donde se levantan las aplicaciones.

El clúster permite crear o destruir contenedores según las necesidades del sistema mediante una máquina maestra, que redirecciona las peticiones a los nodos según la carga de cada uno. Gracias a esto se permite mantener una flexibilidad en cuanto a las capacidades del sistema, escalándolo cuando sea necesario.

4.6 CI/CD y Jenkins

CI/CD son las siglas en inglés de “Continuous Integration” o “Integración Continua”, “Continuous Delivery” o “Entrega Continua” y “Continuous Deployment” o “Despliegue Continuo”.

La integración continua consiste en combinar los cambios de código de los miembros del equipo de desarrollo de forma periódica, ejecutando tras este proceso pruebas automáticas. Gracias a esto se consigue que la combinación de los cambios sea mucho más sencilla y se produzcan menos conflictos, permitiendo proporcionar al cliente actualizaciones con mayor rapidez.

La entrega continua e implementación continua consisten en preparar y subir automáticamente los cambios de código a los entornos seleccionados, realizando las

pruebas necesarias tras la compilación para comprobar el correcto funcionamiento de la subida. La diferencia entre una y otra reside en que la entrega continua necesita de aprobación manual para realizar una actualización en producción.

En definitiva, estas prácticas consisten en automatizar los procesos de integración y despliegue con el fin de mejorar la combinación del trabajo de los desarrolladores y reducir el tiempo y los errores humanos en la realización de las subidas.

Jenkins es un servidor open source que facilita, entre otras cosas, la automatización de todo tipo de tareas en el desarrollo de software, desde la construcción de proyectos y la realización de sus pruebas hasta la entrega y el despliegue de ese software [38].

Para entender Jenkins, lo más sencillo es explicar un ejemplo típico de un pipeline o recorrido CI/CD. Jenkins, a través de la configuración establecida para cada proyecto, automáticamente, obtiene el código del proyecto del sistema de control de versiones, en nuestro caso Git. A continuación, se compila el proyecto y se ejecutan las pruebas unitarias automáticamente. El siguiente paso es realizar las pruebas de calidad y de seguridad, este paso en nuestro caso se realiza con SonarQube. En caso de que las pruebas se hayan pasado correctamente, se realiza el despliegue en un entorno que no es el de producción, donde se pasan las pruebas automáticas de integración. Una vez hecho este paso se realiza la subida definitiva al entorno de producción. Todas estas etapas o stages se pueden configurar independientemente para cada proyecto, añadiendo o quitando las que sean necesarias. En caso de que alguna de las etapas falle, se paraliza el proceso y avisa a los responsables acerca del error.

4.7 Git

Es un sistema de control de versiones gratuito y de código abierto para el desarrollo de software, lanzado inicialmente en 2005 por Linus Torvalds. [18]

Permite trabajar simultáneamente sobre archivos compartidos entre los miembros de un equipo, utilizando diferentes ramas de desarrollo y manteniendo un registro de cambios. Gracias a esto se consigue una alta velocidad y eficiencia en el desarrollo, además de una buena gestión de los conflictos que se producen en el código al trabajar varias personas sobre el mismo.

Tiene muchas ventajas con respecto a otros softwares de control de versiones, como Subversion, Team Foundation o Mercurial. Git permite tener un repositorio local y uno remoto, permitiendo trabajar sin conexión de forma local, y actualizando los cambios con respecto al repositorio remoto una vez se disponga de la conexión.

Gracias al uso de Git, se ha podido seguir un flujo de trabajo no lineal, añadiendo cambios y recuperando los necesarios de otras ramas de desarrollo. Sin su uso, no se habría podido desarrollar el proyecto con metodologías ágiles.

El flujo de trabajo seguido en el proyecto es similar al propuesto por Vincent Driessen [39]. En este flujo las ramas principales son la rama master, donde se encuentra el código subido a producción y la rama principal de desarrollo llamada develop.

A partir de la rama develop se crean las ramas de features, una por cada funcionalidad nueva. Una vez se dispone de una versión estable en desarrollo se crea una rama release, la cual se sube al entorno de integración y se convierte en candidata a ser pasada a producción, donde se resuelven pequeños bugs que se encuentren. Cuando se pasa la aplicación a producción, si se necesita resolver una incidencia, se abre una rama hotfix, y una vez resuelta, se vuelve a integrar en master y en develop.

En la siguiente página se muestra un esquema del flujo de Git que estamos siguiento, propuesto por Vincent Driessen, además de una captura del cliente de Git, GitKraken [40], donde se muestra el gráfico de ramas de uno de los repositorios del proyecto.

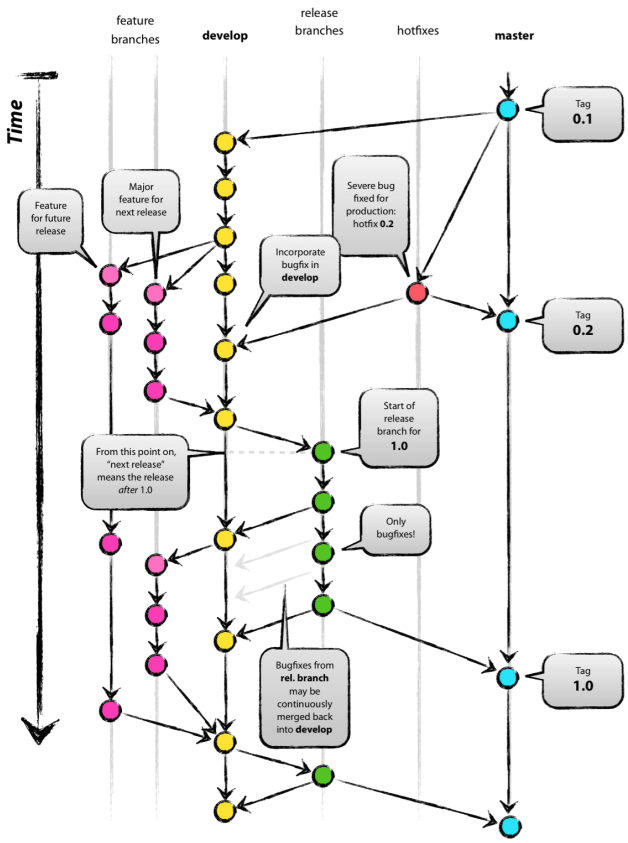

Ilustración 13: Flujo de trabajo de Git de Vincent Driessen [39]

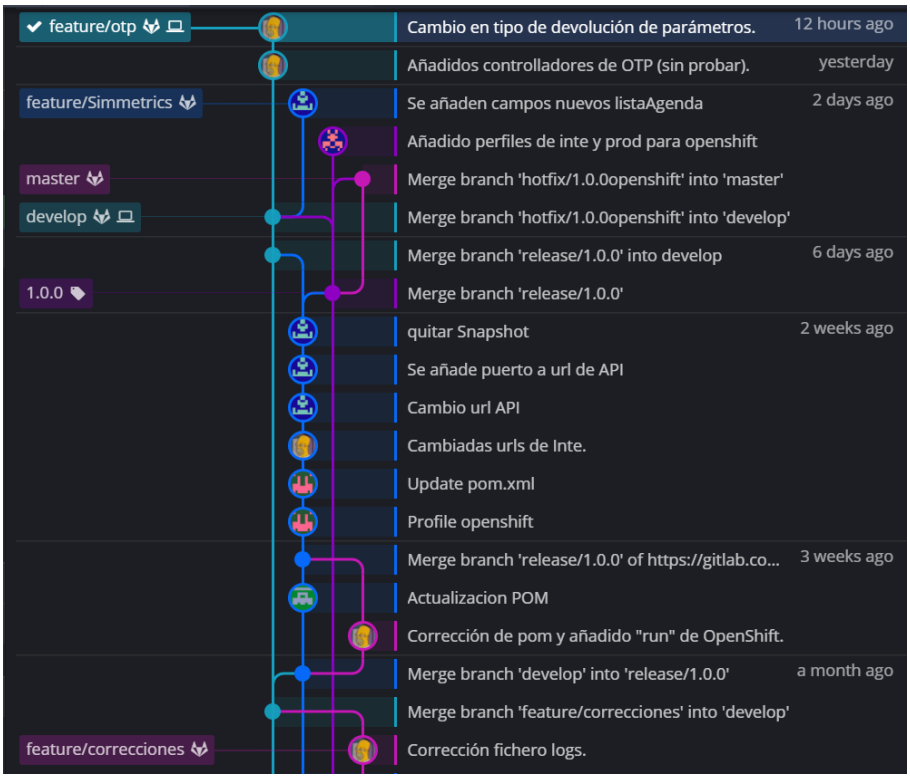

Ilustración 14: Flujo de trabajo de Git de repositorio “Modo delegado back-end”

5 Integración y pruebas

5.1 SonarQube

SonarQube es un software de código abierto diseñado para realizar análisis del código de una aplicación, con el fin de mejorarlo y solucionar problemas [41].

Permite realizar el análisis de múltiples lenguajes, Java, C#, Python, HTML, CSS, TypeScript, entre otros, obteniendo un informe con los resultados del análisis. Gracias a esto, se consigue encontrar código duplicado o con demasiada complejidad, código que no se ajusta a los estándares, bugs, falta de comentarios y fallos de seguridad.

Proporciona una integración con varias herramientas de gestión y construcción de proyectos como Maven, Gradle o Ant y herramientas de integración continua como Jenkins, GitLab CI, Travis CI o Bamboo.

Gracias a SonarQube, hemos podido mejorar la calidad de nuestro código y descubrir vulnerabilidades que tenía nuestra aplicación, permitiendo corregirlas antes de realizar el paso a producción.

5.2 Pruebas unitarias

Las pruebas unitarias son uno de los pilares clave en cualquier proyecto de desarrollo de software. Consiste en realizar pruebas de una única funcionalidad para un conjunto de condiciones para comprobar que el funcionamiento es el esperado.

Gracias a ellas, mejora la calidad del software y aumenta la productividad del desarrollo. A pesar del tiempo que consume el desarrollo de las pruebas unitarias, supone un ahorro en el futuro, tiempo que se emplearía en depurar el código en busca de fallos.

Existen numerosas herramientas y frameworks para la automatización y creación de pruebas unitarias. En el proyecto de HCO se han utilizado Junit [42], Mockito [43] y Powermock [44] para la realización de las mismas.

Como se ha mencionado anteriormente, JUnit se trata de un framework para la realización de pruebas unitarias de aplicaciones Java. Para cada prueba, según la entrada, se comprueba el valor de salida de esta, en caso de que el resultado sea el esperado la prueba será exitosa, en caso de que el resultado no lo sea, la prueba será errónea.

Además de JUnit, se han empleado dos frameworks de Java, Mockito y una extensión de este, PowerMock, que amplían las funcionalidades de las pruebas unitarias de JUnit, permitiendo crear objetos mock.

Los objectos mock, también llamados objetos simulados, imitan de forma controlada el comportamiento de objetos reales, cuando es impracticable usar estos para realizar las pruebas. Permiten devolver resultados determinados, independientemente de la complejidad del objeto real.

En este proyecto, se han usado mocks para simular las respuestas de llamadas a las APIs de agendas, la simulación del servidor de correo SMTP o determinadas consultas a base de datos. [45] [46]

5.3 Pruebas de integración

Las pruebas de integración se realizan una vez se han pasado las pruebas unitarias, para comprobar que todas las funcionalidades que se han probado individualmente funcionan también de forma conjunta.

Existen varios tipos de pruebas de integración, las de caja negra, las de caja blanca y las de caja gris. En las pruebas de caja negra, la persona que las realiza no conoce la estructura interna de la aplicación. En las pruebas de caja blanca, conoce la implementación de la aplicación e introduce las entradas adecuadas comprobando que los caminos recorridos y las salidas son los esperados. Por último, en las pruebas de caja gris se conoce parcialmente la estructura interna del programa.

En este proyecto se han realizado pruebas de caja blanca, en las que cada miembro del equipo ha probado el ciclo completo de las funcionalidades que se han desarrollado. Además, se han realizado pruebas de caja negra en la integración de las agendas médicas, de las que no se tenía conocimiento ni de su diseño ni de estructura interna.

Disponemos de un cuaderno de pruebas que se ha completado y actualizado con cada nueva funcionalidad. Por cada una de ellas, se han anotado las posibles entradas y sus correspondientes salidas esperadas, además del resultado real obtenido tras realizar la prueba.

Las pruebas de integración se han realizado de varias maneras, mediante el uso de herramientas de testing de APIs como Postman, realizando pruebas en Java para comprobar que los flujos completos funcionan correctamente y se obtienen los resultados esperados y las mediante la navegación a través de la interfaz web del “Modo delegado”.

Las pruebas de API son un eje fundamental de las pruebas de integración. Un funcionamiento incorrecto implica también un funcionamiento incorrecto de la interfaz web, ya que ésta que consume sus servicios. Además, el mantenimiento de la automatización de pruebas de una API es mucho más sencillo que el de la interfaz web, debido a la existencia de más herramientas y a los cambios frecuentes en el desarrollo ágil de software.

Postman se trata de la herramienta para la realización de pruebas de la API que se ha utilizado en el proyecto. Permite realizar y automatizar peticiones HTTP para comprobar que todos los endpoints ofrecidos funcionan correctamente.

Para comprobar el correcto funcionamiento de la interfaz gráfica se han realizado pruebas de forma manual mediante el “Modo delegado”. Se han probado ciclos completos de todas las funcionalidades ofrecidas, observando que todos los componentes funcionaban en conjunto correctamente. [47]

6 Conclusiones y trabajo futuro

6.1 Conclusiones

En cuanto a las conclusiones del proyecto, esta aplicación es una solución al problema de estandarización de las agendas y aseguradoras médicas, permitiendo a estas últimas solicitar los servicios de las agendas a través de un único punto de acceso, sin necesidad de realizar conexiones individuales con cada una de ellas.

Supone trabajar para múltiples clientes, las grandes aseguradoras, como Adeslas, IMQ, Agrupaciò o Caser, entre otras, con el reto inicial de diseñar el proyecto para captar a estos clientes.

A pesar de que la aplicación se ha lanzado exitosamente a producción, existen algunas mejoras que se añadirán en futuras versiones, como el emparejamiento automático de agendas en lugar del sistema de traducción.

En cuanto a las conclusiones personales, este proyecto ha sido un reto profesional y académico, que me ha permitido aprender y evolucionar en numerosas áreas, trabajando en un entorno real con tecnologías punteras.

He podido participar en la elaboración de los documentos de análisis y diseño, escribiendo casos de uso y diseñando diagramas de clases, de secuencia y de arquitectura.

He mejorado mis conocimientos de Java, aprendiendo a usar Spring Framework, Maven, servicios web, así como JDBC para el acceso a bases de datos relacionales SQL. Además, he aprendido a estructurar correctamente un proyecto, siguiendo los principios SOLID para el diseño de software robusto y mantenible.

He adquirido la capacidad de desarrollar aplicaciones con el framework de front-end Angular. Cómo estructurar proyectos y utilizar componentes externos, cómo consumir servicios mediante peticiones HTTP, cómo diseñar vistas utilizando CSS, SASS, Angular Material y Bootstrap.

Además, he podido aprender el funcionamiento y las herramientas de computación en la nube, en concreto, parte de las herramientas de AWS, Docker, Kubernetes, Jenkins, Red Hat Openshift o los repositorios Nexus para mantener versiones de librerías, tecnologías punteras, que están cada vez más demandadas en el mercado laboral.

Todas estas herramientas y tecnologías en conjunto han supuesto una notable mejora en cuando a mis conocimientos de la informática y de las tecnologías de la información.

6.2 Trabajo futuro

Este proyecto abre muchas puertas para realizar trabajos futuros, alguno de los cuales ya está en desarrollo.

Una de las funcionalidades nuevas que se están añadiendo es el matching de datos de agendas, centros y facultativos de forma automática. Esto se está realizando mediante "empajeramiento aproximado de cadenas" o "approximate string matching" en inglés, conocido también como "fuzzy string matching". Esta es una técnica para encontrar cadenas de texto que son similares entre sí, pero no idénticas [48].

La finalidad de esto es que, en lugar de realizar las traducciones de códigos de forma manual, se realicen mediante la comparación de cadenas de caracteres, consiguiendo obtener los datos de manera automática. Para realizar esto se está utilizando SimMetrics [49], una librería de Java de código abierto para calcular distancias de Levenshtein y similitudes de coseno entre dos cadenas de texto.

Otra de las nuevas implementaciones es la creación de una caché mediante Redis [50], un RDS (Remote Dictionary Server) de código abierto para mantener estructuras de datos en memoria. Su uso está destinado a almacenar los datos de las agendas que se emparejen de forma automática mediante el proceso anteriormente descrito de comparación de cadenas.

Por último, tendrá lugar el desarrollo de un proyecto nuevo, cuya finalidad será la de elaborar un cuadro médico propio de la compañía, cuyos datos se utilicen para realizar las comparaciones necesarias para la funcionalidad de "fuzzy string matching".

Referencias

[1] Qué es SCRUM, https://proyectosagiles.org/que-es-scrum
[2] Página web de Taiga.io, https://taiga.io/
[3] Alberto Vigario, "La sanidad privada deja de ser un bien de lujo en España", https://www.eleconomista.es/empresas-finanzas/noticias/9548133/11/18/La-sanidad-privada-deja-de-ser-un-bien-de-lujo-en-Espana.html
[4] Expansión, "La sanidad privada mueve más de 30.000 millones de euros al año en España", http://www.expansion.com/empresas/2018/03/23/5ab4022d468aeba7208b464b.html
[5] El País, Departamento de Infografía, "Las cuentas de la salud", https://elpais.com/elpais/2019/05/02/media/1556809379_823938.html?rel=mas
[6] Ley 6/2018, de 3 de julio, de Presupuestos Generales del Estado para el año 2018. Boletín Oficial del Estado, 4 de julio de 2018, núm. 161, páginas 66621 a 67354. Disponible en https://www.boe.es/eli/es/l/2018/07/03/6
[7] Agencia EFE, Europa Press, "Los Presupuestos 2018 aumentan el gasto en Sanidad un 3,9% y el de Educación un 3%", https://www.eleconomista.es/economia/noticias/9044344/04/18/Los-Presupuestos-2018-aumentan-el-gasto-en-sanidad-un-39-y-el-de-educacion-un-3.html
[8] Gaceta Médica, "El Gobierno destina 4.251 millones a Sanidad en los Presupuestos Generales del Estado de 2018", https://www.gacetamedica.com/newsletter/gaceta-medica/el-gobierno-destina-4-251-millones-a-sanidad-en-los-presupuestos-generales-del-estado-de-2018-CL1481127
[9] María Fernández, "En España la salud es cada vez más privada", https://elpais.com/economia/2019/05/03/actualidad/1556897877_211385.html
[10] elEconomista, "Los seguros sanitarios en España viven sus mejores años", https://www.eleconomista.es/salud-innovacion/noticias/9097463/04/18/Los-seguros-de-salud-en-Espana-viven-sus-mejores-anos.html
[11] Deloitte, "El gasto global en asistencia sanitaria crecerá más del 4% hasta 2019", https://www2.deloitte.com/es/es/pages/life-sciences-and-healthcare/articles/estudio-global-del-sector-sanitario.html
[12] Mateo Rivalta, "Forbes Summit Healthcare: El futuro del seguro de salud", http://forbes.es/business/39653/forbes-summit-healthcare-futuro-del-seguro-salud/
[13] Página web de TVS y TVS+, https://tvs.chipcard-salud.es
[14] Página web de Java EE, https://www.oracle.com/technetwork/java/javaee/overview/index.html
[15] Angular, https://angular.io/
[16] Red Hat Openshift, https://www.openshift.com/
[17] AWS (Amazon Web Services), https://aws.amazon.com/
[18] Git source-control management, https://git-scm.com
[19] Amazon Aurora, https://aws.amazon.com/es/rds/aurora/
[20] Amazon DynamoDB, https://aws.amazon.com/dynamodb/
[21] Spring Framework, https://spring.io/
[22] Apache Maven, https://maven.apache.org/
[23] Telosys, https://www.telosys.org/
[24] Eclipse, https://www.eclipse.org/
[25] TypeScript, https://www.typescriptlang.org/
[26] SASS, https://sass-lang.com/

[27] Neoteric, “Single-page application vs. multiple-page application”, https://medium.com/@NeotericEU/single-page-application-vs-multiple-page-application-2591588efe58
[28] Node.js, https://nodejs.org
[29] Angular Material, https://material.angular.io/
[30] Material Design, https://material.io/
[31] Bootstrap, https://getbootstrap.com/
[32] Amazon S3, https://aws.amazon.com/s3/
[33] Amazon EC2, https://aws.amazon.com/ec2/
[34] OpenShift, https://www.openshift.com/
[35] Docker, https://www.docker.com/
[36] Kubernetes, https://kubernetes.io/
[37] Web de Redsys España, http://www.redsys.es/
[38] Jenkins, https://jenkins.io/
[39] Vincent Driessen, “A successful Git branching model”, https://nvie.com/posts/a-successful-git-branching-model/
[40] Cliente de Git GitKraken, https://www.gitkraken.com/
[41] SonarQube, https://www.sonarqube.org/
[42] JUnit, https://junit.org
[43] Mockito, https://site.mockito.org/
[44] GitHub de PowerMock, https://github.com/powermock/powermock
[45] Andrew Spencer, “Unit Testing Fundamentals”, DZone, https://dzone.com/articles/unit-testing-fundamentals
[46] Michael T. Minella, “JUnit and EasyMock”, DZone Refcardz
[47] Software Testing Fundamentals, “Integration Testing”, http://softwaretestingfundamentals.com/integration-testing/
[48] Julien Tregoat, “An Introduction to Fuzzy String Matching”, https://medium.com/@julientregoat/an-introduction-to-fuzzy-string-matching-178805cca2ab
[49] GitHub de SimMetrics, https://github.com/Simmetrics/simmetrics
[50] Redis, https://redis.io/

Glosario

HCO	Hub de Citación Online
Hub	Concentrador de elementos
Modo delegado	Front-end utilizado ofrecido a las aseguradoras como alternativa a una parte frontal propia de cada compañía
API	Application Programming Interface
REST	Representation State Transfer, Transferencia de Estado Representacional
TTM	Time to market
MVP	Minimum Viable Product, Mínimo producto viable
SPOF	Punto simple de fallo (single point of failure)
Front-end	Capa de presentación mostrada en el lado del cliente
Back-end	Capa que se encuentra en el lado del servidor, que sirve al front-end, encargándose de la manipulación de datos.
AWS	Amazon Web Services
IoT	Internet of Things, Internet de las cosas
IaaS	Infrastructure as a Service, Infraestructura como Servicio
PaaS	Platform as a Service, Plataforma como Servicio
GUI	Graphical user interface, Interfaz gráfica de usuario
OTP	One-time password

Anexos

A Capturas de pantalla

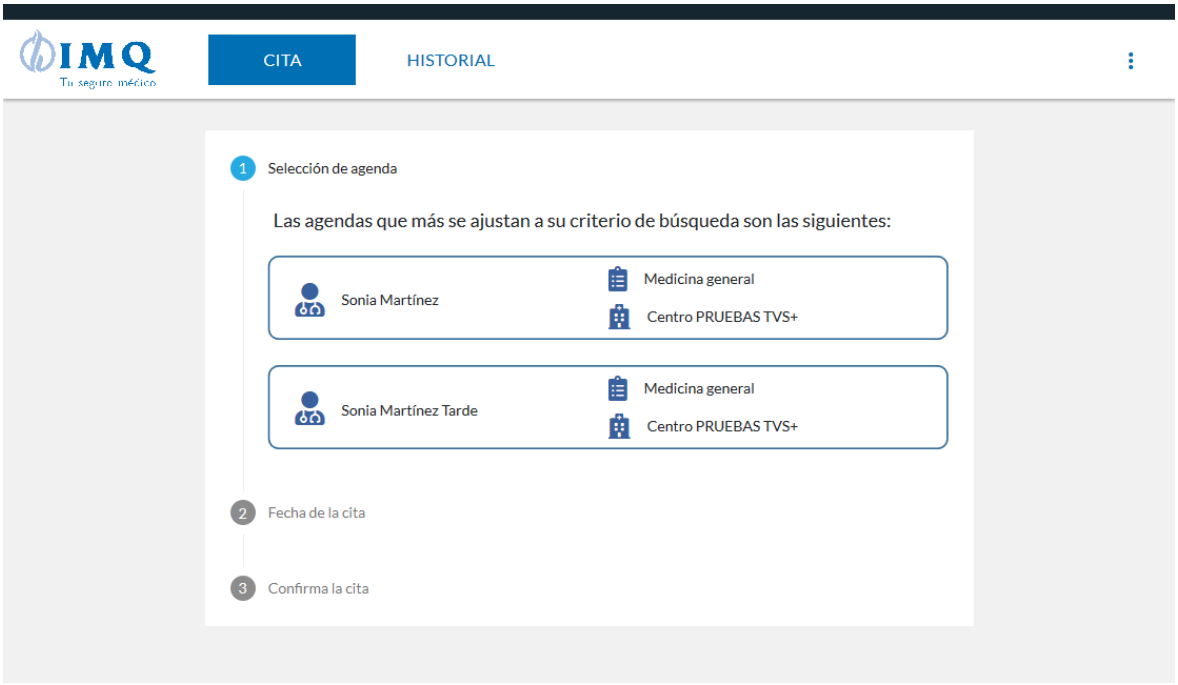

Ilustración 15: Selección de agenda con estilo IMQ

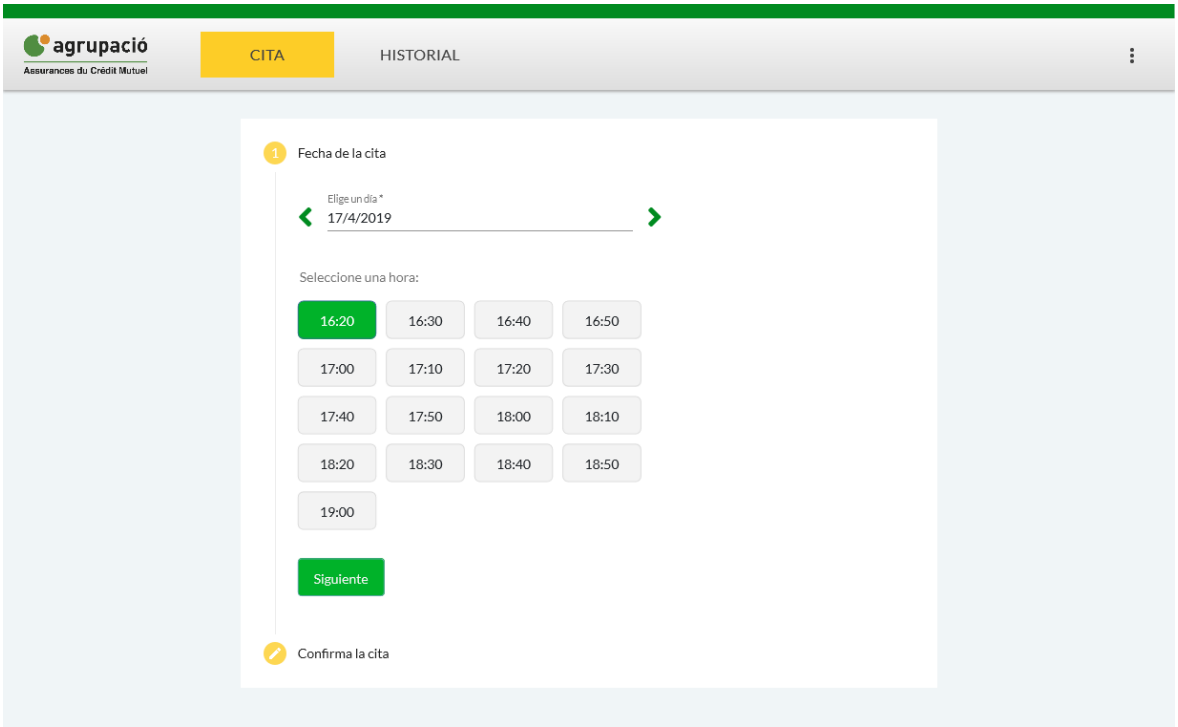

Ilustración 16: Solicitud de cita con estilo de Agrupaciò

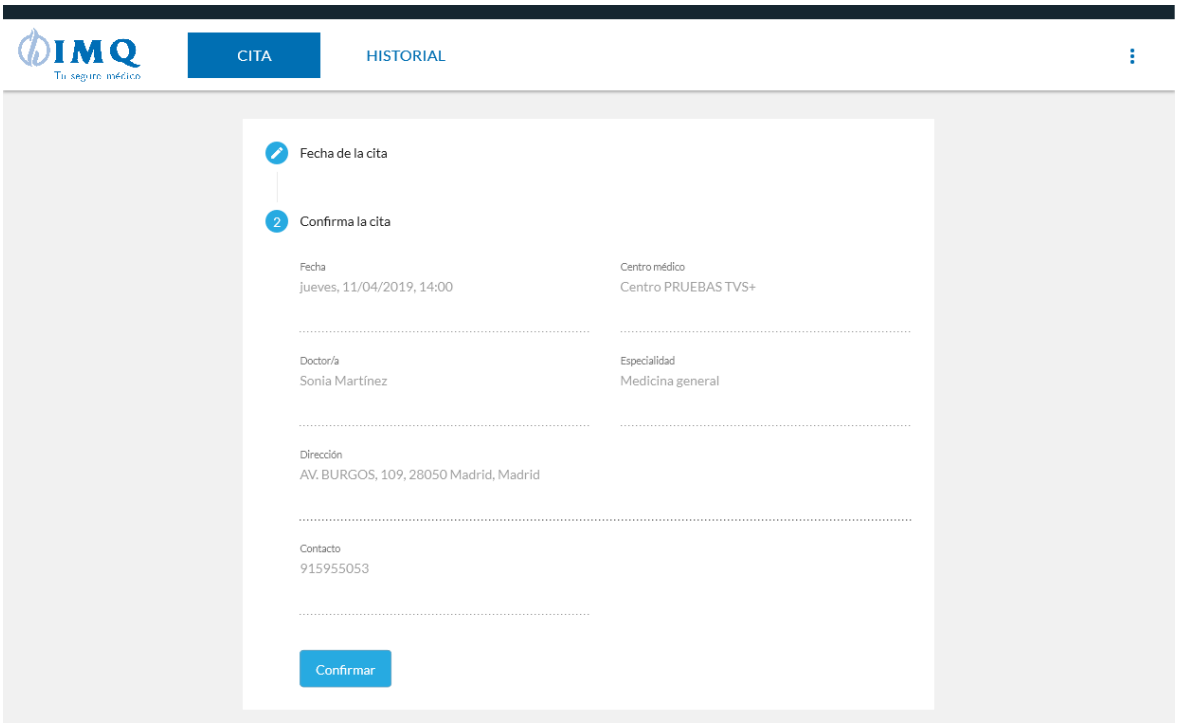

Ilustración 17: Solicitud de cita con estilo de IMQ

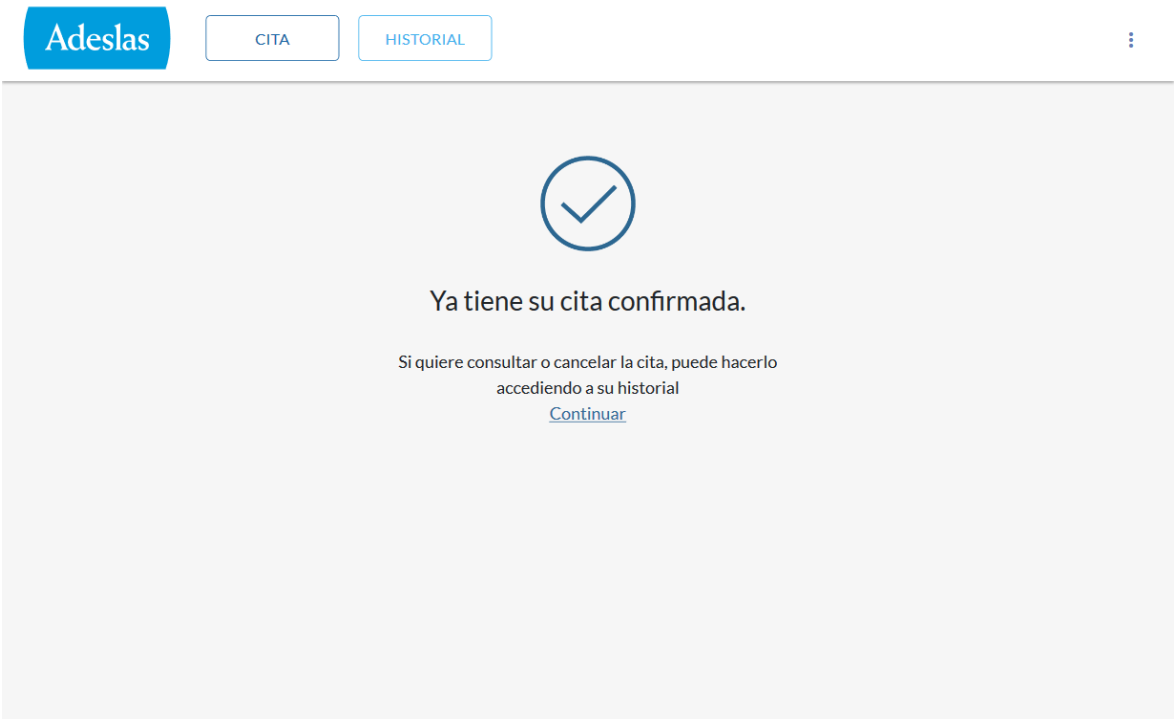

Ilustración 18: Cita confirmada de "Modo delegado" con estilo de Adeslas

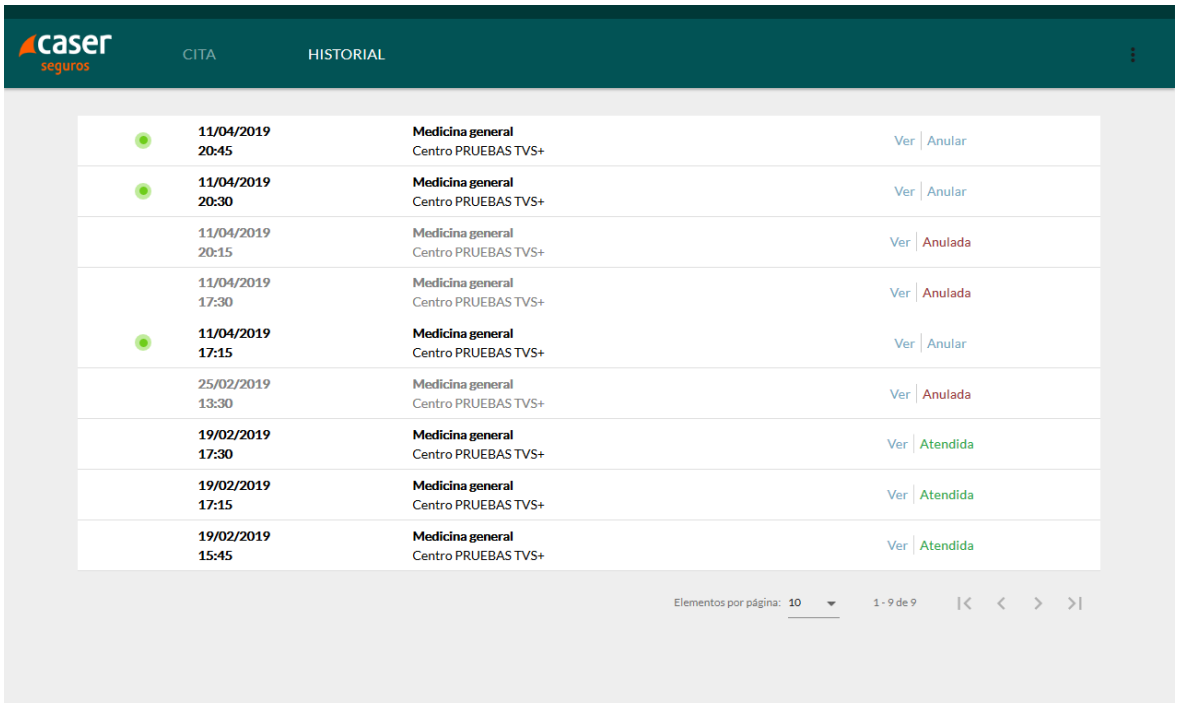

Ilustración 19: Historial de citas con estilo de Caser

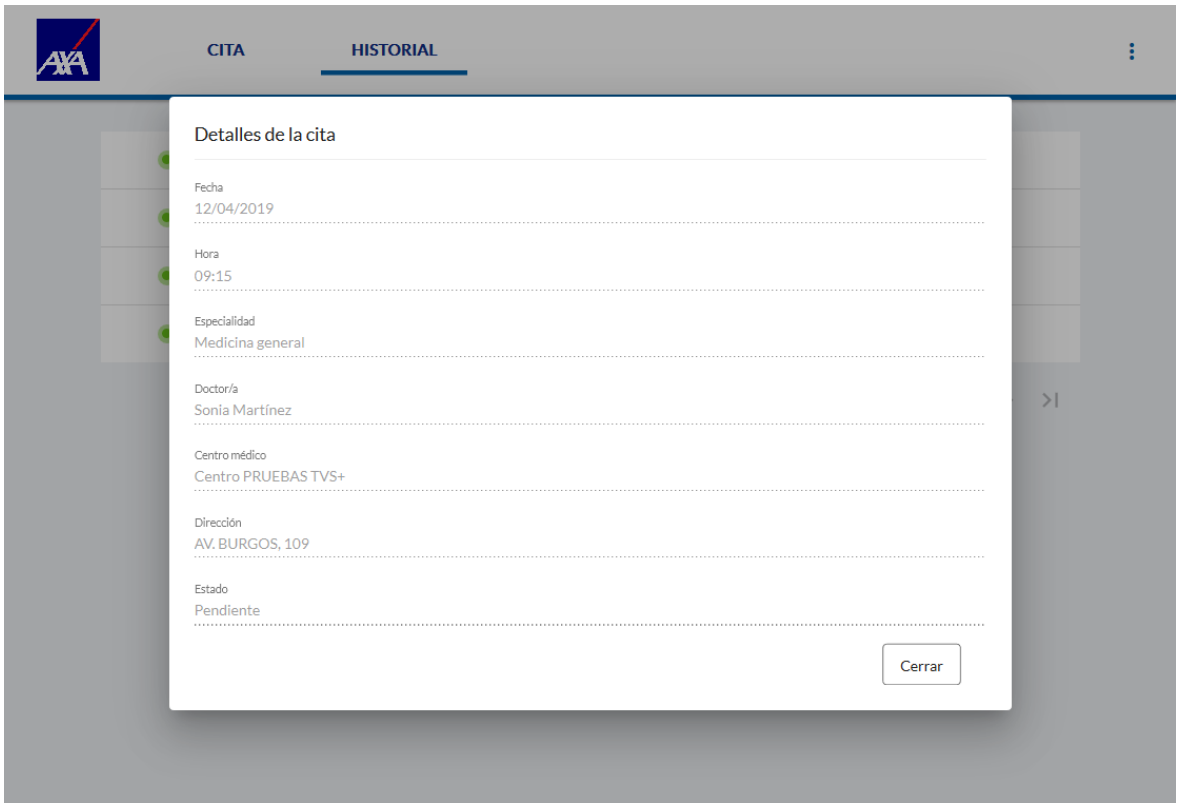

Ilustración 20: Detalles de cita con estilo de AXA

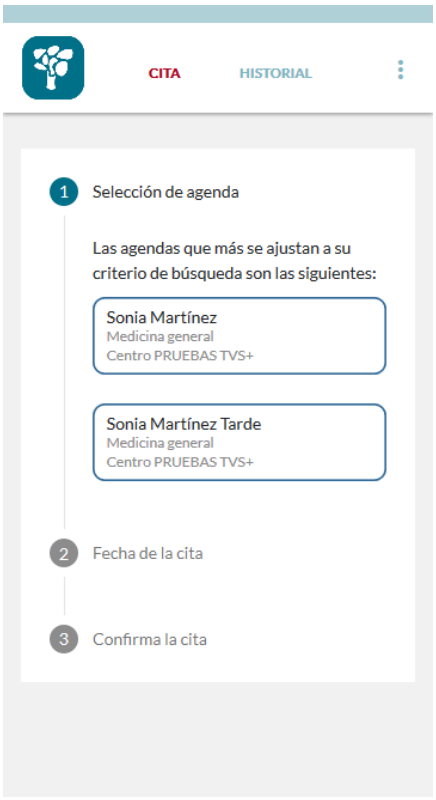

Ilustración 21: Selección de agenda con estilo Mutua General de Cataluña modo móvil

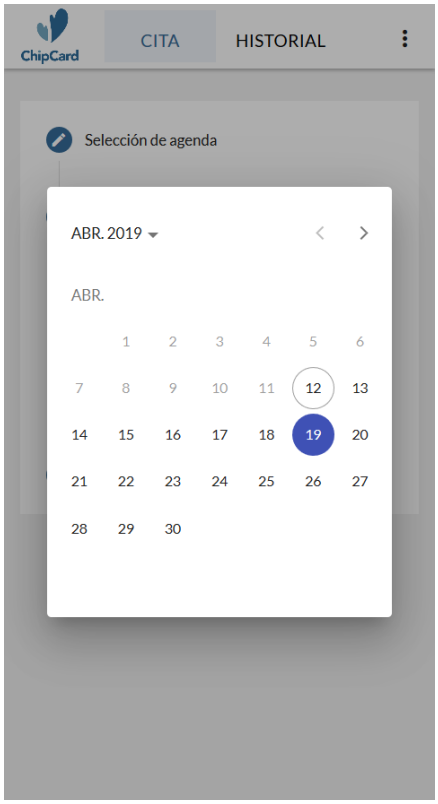

Ilustración 22: Selección de fecha con estilo ChipCard modo móvil

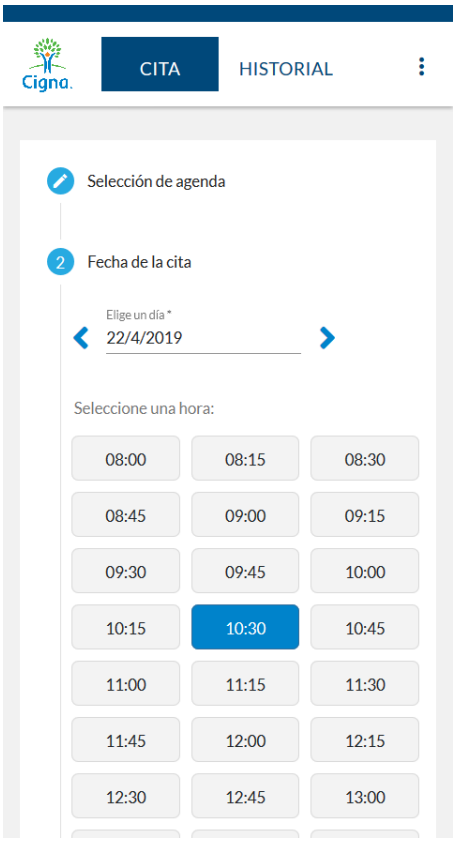

Ilustración 23: Selección de horarios con estilo de Cigna modo móvil

Ilustración 24: Historial de citas con estilo de Agrupaciò modo móvil