

TRABAJO FIN DE GRADO

APLICACIÓN MÓVIL PARA RECABAR LA OPINIÓN DE LOS PACIENTES SOBRE LA CALIDAD DEL SERVICIO SANITARIO PERCIBIDO

TRABAJO FIN DE GRADO PARA
LA OBTENCIÓN DEL TÍTULO DE
GRADUADO EN INGENIERÍA EN
TECNOLOGÍAS INDUSTRIALES

Junio 2019

Miguel Cordero Collar

DIRECTORES DEL TRABAJO FIN DE GRADO:

Joaquín Ordieres Meré
Carlos Elvira Martínez

AGRADECIMIENTOS

Este trabajo significa la culminación de una etapa muy importante de mi vida, llena de altibajos, para ser sinceros, esta carrera es lo más duro a lo que me he enfrentado, y en el camino he aprendido mucho, tanto conocimientos técnicos, como para la vida, pero sobretodo me he conocido a mí mismo un poco mejor.

Me veo en la obligación de agradecer a muchas personas, empezando por mis padres y hermana, quienes me han apoyado incondicionalmente, animado y confiado en mí.

A todos aquellos amigos del Colegio Mayor Alcalá, que me hayan ayudado, apoyado o animado alguna vez, más concretamente me gustaría agradecer a Alberto, Pepe, Roberto, Arturo, Cabrera, Guado y Marcos.

A mis amigos y compañeros de carrera y sufrimiento, concretamente a Javier, Alfonso, Maripi, Oscar, Miguel Ángel y Javi Vera.

A mis amigos de Málaga, sobre todo a Javi y Patri, con quienes he compartido días de biblioteca y cada vez que volvía me hacían sentir de volvía a casa.

En relación al TFG, muchas gracias a los dos tutores, Don Joaquín Ordieres Meré y Don Carlos Elvira Martínez, destacando su amabilidad, ganas de hacer bien las cosas y disponibilidad durante todo el proyecto.

Y por último a mis abuelos, tanto paternos como maternos, sobre todo por la educación y oportunidades que brindaron a mis padres para que ellos pudieran hacer lo mismo conmigo.

RESUMEN

El presente Trabajo de Fin de Grado busca el diseño y desarrollo de un prototipo de aplicación móvil para el Hospital Clínico San Carlos de Madrid, con la finalidad de crear un canal de comunicación directo entre el paciente y los responsables de diferentes servicios dentro del hospital.

Actualmente, muchas empresas e instituciones se encuentran inmersas en la llamada “Transformación Digital”, que consiste en la reorganización de los métodos de trabajo y estrategias en general, para obtener más beneficios gracias a la digitalización de los mismos y la implementación dinámica de nuevas tecnologías. Este TFG estudia la posibilidad de implantación de una herramienta digital, en forma de aplicación, para continuar con dicha transformación.

La idea para la realización de este proyecto fue del doctor Carlos Elvira, Jefe del Servicio de Admisión y Documentación Clínica, quien al conocer de primera mano la organización del hospital, determinó que el desarrollo de esta aplicación sería de gran valor tanto para los pacientes como para el hospital.

El Hospital Clínico San Carlos es un centro hospitalario de titularidad pública, situado en la calle del Profesor Martín Lagos s/n. CP 28040, en el distrito de Moncloa de la ciudad de Madrid. Está administrado por el Servicio Madrileño de Salud, dependiente de la Consejería de Sanidad de la Comunidad de Madrid.

Los orígenes del Hospital Clínico San Carlos se remontan a 1787 y se encuentra entre los más reputados hospitales públicos de España, tanto a nivel de innovación, investigación, y asistencia hospitalaria, prestando servicios a unas de 400.000 ciudadanos en su área de influencia.

Actualmente el Hospital no cuenta con este servicio de aplicación móvil propia, que supone la pérdida de oportunidad de mejora del servicio, efectividad y eficiencia, pues las aplicaciones móviles son un gran recurso para cualquier organización al objeto mejorar muchos aspectos de las mismas, y caminar hacia la excelencia en el servicio al paciente y a su entorno familiar.

El prototipo ha sido desarrollado usando el framework Cordova Apache, el cual permite la programación y diseño de aplicaciones haciendo uso de lenguajes de desarrollo web, como son HTML5, CSS3, y JavaScript, con todos los beneficios que esto conlleva, que se explicitan y mencionan en los diferentes capítulos de referencia. Uno de los puntos positivos que tiene este framework es que permite el desarrollo simultáneo para varias plataformas, como son Android, iOS, Windows Phone y web. A parte de la parte web de la aplicación, Cordova permite hacer uso de las capacidades propias de las diferencias plataformas por medio de plugins.

Con el objetivo de prototipar todas las partes necesarias para el correcto funcionamiento de la aplicación, se ha diseñado una base de datos para poder realizar todas las funciones establecidas, esta, en el caso de llevarse a la práctica, debería conectarse a la base de datos del hospital para obtener los datos reales de los pacientes del hospital. Se han introducido datos ficticios, pero de apariencia real con el objetivo de realizar pruebas y comprobar el buen funcionamiento de todos los elementos. La gestión de la base de datos en cuestión se realiza a través de PhpMyAdmin, un gestor gratuito y código abierto. Por último, la comunicación entre la base de datos y la aplicación se

realiza a través de consultas AJAX, que tienen la ventaja de ser asíncronas y simples de implementar por ambos extremos de la comunicación.

El diseño de la aplicación se realizó con el objetivo de ser intuitiva y fluida, se ha intentado cuidar la presentación y estética, pero sería conveniente valorar que, a la hora de implementarla hacer una revisión de esta estética por un profesional, ya que es uno de los puntos más importantes para la rápida adaptación por parte del usuario. La aplicación cuenta una barra de navegación permanente para acceder a las diferentes secciones, a la vez que se puede acceder a estas últimas por medio de un menú principal con iconos.

El prototipo cuenta con cuatro secciones principales, además de estas se ha incluido una página de inicio de sesión y el menú principal comentado anteriormente, aquí una explicación más en detalle de cada uno:

Inicio de sesión: aquí, el usuario inicia sesión, mediante una serie de parámetros verificamos que se encuentra ingresado o en urgencias, esta información se contrasta con la base de datos, y en caso afirmativo se le da acceso a la aplicación.

Menú principal: se le presenta al usuario con un mensaje de bienvenida y una breve explicación, también tiene acceso a través de iconos a las otras secciones de la aplicación.

Incidencias: en esta sección, el usuario puede reportar sus incidencias, eligiendo dentro de unas predeterminadas o escribir una personalizada. También tiene la opción de acceder a una tabla con el historial de incidencias y su estado de desarrollo, si han sido solucionadas o si siguen abiertas.

Encuestas: aquí el usuario elige la encuesta que quiere realizar entre varias opciones dependiendo de su perfil y una vez que ha seleccionado una, la realiza y es enviada a la base de datos.

Servicios: el usuario debe seleccionar el servicio que desea solicitar, algunos, como el de biblioteca o comidas, requiere selección de opción dentro del servicio, además, puede escribir un texto explicativo de su petición si así lo desea. Al igual que las incidencias, el usuario puede acceder desde la aplicación al historial y estado de los servicios solicitados.

Ajustes: es la pantalla desde la que se cierra la sesión, además puede revisar su información y controlar las notificaciones.

Tal y como se ha mencionado anteriormente, se realizan notificaciones al usuario, con la finalidad de animarle a cumplimentar encuestas para compartir su opinión. Esta información queda almacenada con el objetivo de analizar los procedimientos y poder optimizar el ratio de participación en las encuestas.

Tanto en el caso de las incidencias como de los servicios, una vez que llega la información a la base de datos se reporta a la persona responsable de realizar el servicio o solucionar la incidencia, creando un canal de comunicación que aporta una mejora de eficiencia. Además, una vez que se cierra una incidencia o servicio queda reflejado en la base de datos y tanto el trabajador como el paciente pueden puntuar con una calificación que refleje su grado de satisfacción.

A lo largo de la memoria se han incluido capturas de pantalla y parte del código con el objetivo de hacer más comprensible el desarrollo de la aplicación.

Resumen

Los beneficios de la aplicación son claros, es un eficaz canal de comunicación, tanto para el paciente, por el acceso más sencillo a los servicios y solución de incidencias más eficaz y rápida, como para el hospital, que ahorra en recursos humanos y materiales, a la vez que aporta un mejor servicio a sus pacientes, que es su objetivo. Por último, la posibilidad de análisis de datos a partir de la base de datos aporta un gran valor a la hora de conocer las fortalezas y debilidades de los procedimientos en el hospital.

PALABRAS CLAVE: Aplicación, App, Android, Hospital Clínico San Carlos, base de datos, prototipo, JSON, AJAX, programación, web, Cordova

CODIGOS UNESCO:

120317 Informática.

120318 Sistemas de información, diseño y componentes.

120323 Lenguajes de programación.

330416 Diseño lógico.

ÍNDICE

Agradecimientos	1
Resumen	3
Índice	7
Índice de Ilustraciones	9
Índice de Tablas	11
Capítulo 1: Introducción	13
1.1 Motivación	13
1.2 Objetivos	14
1.3 Estructuración de la memoria	15
Capítulo 2: El Hospital San Carlos	17
2.1 Historia	17
2.2 Misión, visión y valores.....	17
2.3 Análisis de organización de servicios	18
2.4 Análisis de organización de incidencias y encuestas.....	19
Capítulo 3: Tecnologías utilizadas	21
3.1 Proceso de selección	21
3.2 Proceso de aprendizaje.....	22
3.3 Apache Cordova.....	23
3.4 Entorno de desarrollo	24
3.5 Lenguajes de Programación.....	25
3.6 Librerías	26
3.7 Control de versiones	27
3.8 Base de Datos.....	28
Capítulo 4: Diseño y funcionalidades de la aplicación.....	29
4.1 Proceso de Diseño	29
4.2 Funcionalidades	30
4.3 Diagrama de casos de uso	31
4.4 Modelo de la aplicación	32
4.5 Diseño de pantallas.....	33
Capítulo 5: Desarrollo de la aplicación.....	35
5.1 Páginas	35
5.1.1 Página Index.....	35
5.1.2 Inicio de sesión	35
5.1.3 Menú inicial.....	37
5.1.4 Incidencias.....	37

5.1.5 Encuestas	42
5.1.6 Servicios	44
5.1.7 Ajustes	46
5.2 Base de datos	48
5.2.1 Diseño de la BBDD	48
5.2.2 Población de la BBDD.....	51
5.2.3 Comunicación mediante Ajax.....	51
5.2.4 Lógica PHP de la BBDD	52
5.3 Notificaciones	54
5.4 Capturas de pantalla	55
Capítulo 6: Conclusiones y líneas futuras de desarrollo	59
6.1 Valoración de impacto y aspectos de responsabilidad.....	59
6.2 Conclusiones.....	60
6.3 Líneas futuras de desarrollo	61
Capítulo 7: Planificación temporal y presupuesto.....	63
7.1 Estructura de descomposición del proyecto	63
7.2 Diagrama de Gantt	64
7.3 Presupuesto	65
Glosario	67
Autoría de Imágenes.....	69
Bibliografía.....	71
Anexo	75
Anexo I: Código desarrollado	75

ÍNDICE DE ILUSTRACIONES

Ilustración 1: Herramienta dbdiagram.io.....	28
Ilustración 2: Mock-up preliminar de la aplicación	29
Ilustración 3: Casos de uso.....	31
Ilustración 4: Modelo de la aplicación	32
Ilustración 5: Código HTML barra de navegación	35
Ilustración 6: Diagrama de sección Inicio de sesión.....	35
Ilustración 7: Código JavaScript redirección sesión iniciada	36
Ilustración 8: Código JavaScript validación formulario	36
Ilustración 9: Código AJAX Inicio sesión.....	36
Ilustración 10: Diagrama de sección Menú de inicio	37
Ilustración 11: Código JavaScript redirección menú de inicio.....	37
Ilustración 12: Diagrama de sección Incidencias.....	37
Ilustración 13: Código de clases incidencias.....	38
Ilustración 14: Código HTML incidencias	38
Ilustración 15: Código opciones incidencias.....	38
Ilustración 16: Código JavaScript animaciones incidencias.....	39
Ilustración 17: Código JavaScript de validación incidencias.....	39
Ilustración 18: Código HTML de menú confirmación de envío	40
Ilustración 19: Código HTML seguimiento incidencias	40
Ilustración 20: Código AJAX de envío de incidencias.....	40
Ilustración 21: Código AJAX de clase incidencia.....	41
Ilustración 22: Código AJAX de obtención de incidencias del usuario.....	41
Ilustración 23: Código JavaScript de creación de incidencias	41
Ilustración 24: Diagrama de sección Encuestas.....	42
Ilustración 25: Código de clase encuesta.....	42
Ilustración 26: Código AJAX de obtención de encuestas	42
Ilustración 27: Código JavaScript de creación de encuestas.....	43
Ilustración 28: Código HTML de selección de encuestas	43
Ilustración 29: Código JavaScript de presentación de encuesta	43
Ilustración 30: Código AJAX de envío de encuestas	43
Ilustración 31: Diagrama de sección Servicios	44
Ilustración 32: Código HTML de selección de servicio	44
Ilustración 33: Código HTML bloques de servicios.....	44
Ilustración 34: Código JavaScript de visualización de bloques de servicios	45
Ilustración 35: Código de clases y creación de opciones de dieta.....	45
Ilustración 36: Código AJAX de envío de servicio	46
Ilustración 37: Diagrama de sección Ajustes.....	46
Ilustración 38: Código JavaScript de visualización de bloques de ajustes.....	47
Ilustración 39: Código JavaScript de cierre de sesión	47
Ilustración 40: Diagrama de tablas de la base de datos	48
Ilustración 41: Tabla users	49
Ilustración 42: Tabla incidents.....	49
Ilustración 43: Tabla instances_hospitalization	49
Ilustración 44: Tabla instances_emergency	49
Ilustración 45: Tabla notification_data.....	50
Ilustración 46: Tabla services.....	50
Ilustración 47: Tabla survey_models.....	50

Ilustración 48: Tabla surveys	50
Ilustración 49: Código script creación de SQL	51
Ilustración 50: Código de creación notificaciones.....	54
Ilustración 51: Captura de ejemplo de notificación	54
Ilustración 52: Código AJAX de envío de notificaciones.....	55
Ilustración 53: Capturas de sección de Inicio de sesión	55
Ilustración 54: Capturas de sección de Incidencias.....	56
Ilustración 55: Capturas de sección de Encuestas.....	56
Ilustración 56: Capturas de sección de Servicios	57
Ilustración 57: Captura de sección de Ajustes	57
Ilustración 58: Estructura de descomposición del proyecto	63
Ilustración 59: Diagrama de Gantt.....	64

ÍNDICE DE TABLAS

Tabla 1: Tecnologías utilizadas.....	21
Tabla 2: Colores utilizados.....	30
Tabla 3: Imágenes utilizadas	33
Tabla 4: Iconos utilizados	33
Tabla 5: Costes de hardware	65
Tabla 6: Costes de recursos humanos.....	65
Tabla 7: Presupuesto total	66

CAPÍTULO 1: INTRODUCCIÓN

1.1 Motivación

Las nuevas tecnologías nos ofrecen la oportunidad de comunicarnos de manera más eficiente y cómoda, a la vez que facilita el registro, análisis e interpretación de datos con el objetivo de encontrar las mejores soluciones en el contexto de nuestra sociedad contemporánea. Además, más del 80% de la población usa actualmente los teléfonos móviles inteligentes, también conocidos como smartphones, lo que permite una rápida difusión y adaptación de las aplicaciones para el beneficio de todas las entidades y personas involucradas.

El ámbito de la salud en el sector tecnológico está teniendo un gran auge en los últimos años, desde cosas tan complejas como el uso de la inteligencia artificial para interpretar los resultados de las pruebas y realización de cirugías complejas, hasta cosas más simples como la contabilización del consumo e ingesta de calorías para poder llevar una alimentación equilibrada.

Todavía existen grandes ineficiencias en todos los sectores, las tecnologías de la información contribuyen de un modo muy relevante a la minimización de estas, y al avance en el desarrollo de aplicaciones y herramientas al servicio de los ciudadanos, con el objetivo de lograr mayores niveles de eficiencia y de costes. El sector sanitario se encuentra inmerso en estos procesos, y es por esto que se debe hacer un análisis exhaustivo de todos los procesos que tienen lugar dentro de un hospital con el objetivo de facilitar el trabajo al proveedor del servicio, al mismo tiempo que se ofrece un mayor valor al usuario.

El doctor Carlos Elvira, Jefe del Servicio de Admisión y Documentación Clínica ha identificado como una de las mayores ineficiencias que se producían en el hospital, la comunicación entre el paciente (o el entorno familiar del mismo) con los diferentes servicios ofrecidos por el Hospital (biblioteca, peluquería, asistencia religiosa, menús, etc...) y también la presentación de sugerencias, incidencias y opinión. El desarrollo de esta aplicación informática supone una herramienta muy decisiva para dar solución a este tipo de ineficiencias.

El objetivo del trabajo consiste en ofrecer al Hospital Clínico San Carlos un prototipo de una aplicación que realice las funciones comentadas anteriormente y permita estudiar la viabilidad de la misma y ventajas e inconvenientes de los diferentes enfoques posibles.

1.2 Objetivos

El principal objetivo es el desarrollo del prototipo una aplicación para el Hospital Clínico San Carlos de Madrid, con el fin de gestionar incidencias, sugerencias, encuestas y múltiples servicios que se ofrecen a los pacientes, para mejorar la experiencia de los usuarios del hospital y poder obtener datos e información de usuarios sobre la calidad de estos procedimientos, para el desarrollo de estrategias de mejora de los mismos.

El prototipo puede funcionar tanto como una página web, aplicación para Android, iOS y Windows Phone, aunque no se han realizado pruebas en estos dos últimos por motivos de tiempo, pero en principio no debería haber ningún problema a la hora de portar el resultado a estas plataformas.

Actualmente la gestión de incidencias, como sería, por ejemplo, problemas con la temperatura de la habitación, se realiza informando a personal de enfermería, y este, debe acordarse y tener tiempo para comunicárselo a la persona responsable, lo cual es un canal de comunicación poco eficiente y confiable, y es habitual que la información no llegue al responsable. Por esto, al implementar un canal de comunicación directo, dichas incidencias podrán ser resueltas en una mayor brevedad y de manera más eficaz.

Además, la aplicación ha sido diseñada para que el usuario pueda hacer un seguimiento del estado de la incidencia, si el responsable del servicio ha sido informado, si se han realizado actuaciones al respecto, y de si el responsable del servicio ha cerrado la incidencia, así como del resultado de las actuaciones, si ha sido satisfactorio o por el contrario no se ha podido solucionar la incidencia y las causas.

Las encuestas actualmente se realizan en papel, siendo repartidas por el servicio de enfermería. Con la aplicación, se podrán diseñar encuestas a medida con facilidad, crear encuestas más focalizadas en perfiles muy definidos, con el objetivo de obtener información más detallada y al estar informatizadas las respuestas, analizar los datos de manera sencilla y rápida, pudiendo obtener resultados más útiles para la mejora de los servicios, instalaciones, etc...

Adicionalmente, a las encuestas se ha implementado la posibilidad de recibir notificaciones recordando al usuario de realizar las mismas, la respuesta a estas notificaciones será almacenada para analizar la receptividad de los usuarios a estas y poder animarles a aportar su opinión para mejorar el servicio que reciben.

Para la gestión de servicios ofertados a los pacientes hospitalizados (biblioteca, peluquería, quiosco, asistencia religiosa, voluntariado, selección de menú), ocurre algo similar a la gestión de incidencias, el paciente informa al servicio de enfermería, y este es el responsable de hacer llegar la información hasta la persona o departamento que se encarga de dicho servicio. A través de la aplicación se generaría un canal directo entre el usuario y el responsable que ofrece el servicio, en principio a través de mail. Y al igual que las incidencias el usuario podrá hacer un seguimiento del estado del servicio.

1.3 Estructuración de la memoria

La memoria del proyecto se ha estructurado en diferentes capítulos, los cuales se explican a continuación.

En el capítulo 2 comienza con una introducción al Hospital Clínico San Carlos, su historia, misión, visión y valores, y, por último, se realiza un análisis del estado actual de los aspectos relevantes del mismo para el desarrollo del proyecto, como son la organización de los servicios, encuestas y solución de incidencias.

En el capítulo 3 se tratan las tecnologías utilizadas en el desarrollo del prototipo, comenzando por el proceso inicial de selección y aprendizaje de estas. Posteriormente se entra más en detalle en las diferentes tecnologías: el marco de desarrollo Cordova, el IDE usado, los lenguajes de programación, librerías, base de datos y el control de versiones del código.

En el capítulo 4 se entra en profundidad en el proceso de diseño de la aplicación, sus distintas etapas, las funcionalidades requeridas, el diagrama de casos de uso, el modelo de la aplicación para poder visualizar las distintas funcionalidades y por último el diseño de pantallas.

En el capítulo 5 se analiza el desarrollo de la aplicación, primeramente, las distintas páginas y su funcionamiento a nivel de código, también se explican las distintas etapas de desarrollo de la base de datos, las notificaciones y finalmente, se incluyen capturas de pantalla de las distintas páginas de la aplicación.

En el capítulo 6 se estudian varios aspectos relativos al resultado del proyecto, como es la valoración de impacto, conclusiones obtenidas y posibles líneas de desarrollo futuro.

El último capítulo, el 7, es un análisis del proyecto a nivel de descomposición estructural, temporal y una estimación económica del coste del proyecto en sí.

CAPÍTULO 2: EL HOSPITAL SAN CARLOS

2.1 Historia

Sus orígenes se remontan al año 1787, nacido como consecuencia de dos Reales Cédulas por parte de Carlos III, comenzando con actividades docentes, siendo el tercer Real Colegio promovido por el Rey.

En la calle Atocha se encontraba el Hospital Provincial, pero este carecía de posibilidades de desarrollo y ampliación, por ello en 1888 se propuso la construcción de un nuevo hospital. El lugar seleccionado finalmente fue la Huera de Moncloa, comenzando las obras en el año 1932. Se preveía inaugurar en octubre de 1936, pero al estallar la guerra civil el edificio fue afectado, requiriendo una reconstrucción.

El día 20 de enero de 1951 comenzó el traslado de actividades del Hospital de la calle Atocha al edificio actual, terminando el 1 de octubre de 1965. Se inauguró el Hospital adecuándose a todas las cátedras, unidades docentes y departamentos con el objetivo de cumplir tres funciones: investigadora, docente y asistencial.

2.2 Misión, visión y valores

La misión del Hospital Clínico San Carlos según declara en su propia página web es:

" Proporcionar a sus ciudadanos un servicio de atención sanitaria especializada, asegurando los más altos niveles de rapidez de respuesta, calidad y eficiencia, conjuntamente con su compromiso en la formación de nuevos profesionales sanitarios e incorporando la investigación biosanitaria.

Esta labor asistencial, de investigación y educativa la lleva a cabo fortaleciendo la comunicación y coordinación con las diferentes estructuras y niveles asistenciales de la Consejería de Sanidad, así como con la Universidad Complutense de Madrid y otras entidades públicas y privadas."

Su visión sería la siguiente:

Nuestra visión es alcanzar la excelencia, es decir, una Atención Especializada integrada en un proceso de continuidad asistencial con la Atención Primaria, de alta calidad (que satisface las expectativas de los ciudadanos) y que ofrece toda la ciencia y la tecnología disponibles en cada momento (complejidad elevada), adaptándose de forma a los cambios del entorno a través de la formación y de la tecnología, y promoviendo la participación de la Comunidad a la que servimos, para que nuestro Hospital sea ejemplo de eficacia, eficiencia y equidad.

Para ello, el Hospital Clínico San Carlos compromete a sus profesionales en una política de mejora continua de la calidad asistencial, fomentando la motivación, promoción y satisfacción individual de sus trabajadores. Además de ser proveedores de una excelente atención sanitaria, queremos ser capaces de ofrecer esperanza e ilusión a todas aquellas personas que buscan en nosotros esa parte de alivio a su problema de salud. Y por último sus valores."

Por último, los valores que aspira a compartir serían:

1. *“Orientar y centrar sus actuaciones en el paciente.”*
2. *“Cuidar a sus profesionales buscando su participación con los objetivos institucionales.”*
3. *“Innovar e incorporar las oportunidades que brindan las nuevas tecnologías.”*
4. *“Comprometerse diariamente con la mejora continua en la calidad asistencial.”*
5. *“Incorporar los recursos derivados del conocimiento y la investigación.”*
6. *“Tener un hospital sostenible y respetuoso con el medio ambiental.”*
7. *“Buscando la máxima eficiencia.”*

2.3 Análisis de organización de servicios

Actualmente los servicios disponibles y que se han acordado incluir en la aplicación son los siguientes:

- Peluquería: Es un servicio que se encarga de cortar y arreglar el pelo a los pacientes, del cuidado de su propia imagen, importante para la estima personal en momentos delicados de salud.
- Biblioteca: Es un servicio muy importante, para muchas personas la lectura supone el viaje a otras realidades y mundos ajenos a la realidad de la coyuntura del momento, que al ser en estos casos la enfermedad y el dolor, la lectura es fundamental para el bienestar espiritual de muchas personas.
- Quiosco: Se pueden solicitar tanto periódicos como revistas.
- Asistencia religiosa: Hay una capilla en el hospital, y el capellán está a la disponibilidad de las personas ingresadas y de sus familiares.
- Voluntariado: Hay grupos de voluntarios externos al hospital que realizan actividades varias, en función de las patologías de los pacientes en las diversas unidades.
- Servicio de comidas: Las personas pueden elegir la opción de menú que estiman más adecuada dentro de las opciones disponibles por su estado de salud.

Estos servicios están organizados del siguiente modo:

En el caso de los cinco primeros, el método que se sigue es el de informar al servicio de enfermería, y estos deben acordarse de pasar dicha información al responsable, lo que es poco eficaz, eficiente, y no seguro, ya que las enfermeras son personas con una carga de trabajo elevada, muy variada, sujeta a momentos de presión, y de responsabilidades elevadas.

Para el servicio de comidas, el día antes se les entrega un menú (impreso en papel) a los pacientes con las diferentes opciones, y este debe seleccionar la opción que interesa y devolver el impreso, este procedimiento es ineficaz y consume muchos recursos económicos y humanos, se demora mucho en conocer los resultados para prever la realización de los diferentes menús, y no es fiable, ya que es común que haya confusiones y equivocaciones.

2.4 Análisis de organización de incidencias y encuestas

Las incidencias, al igual que los servicios, se informan al servicio de enfermería, que es responsable de transmitir dicha información a la persona encargada de solucionar la incidencia, o en el caso de que sea una incidencia sin solución directa, esta problemática se pierde a nivel estadístico.

Las encuestas se realizan en papel, y es el servicio de enfermería los encargados de entregarlas y recogerlas, y la hora de interpretarlas, al no estar informatizadas consume una gran cantidad de recursos humanos.

CAPÍTULO 3: TECNOLOGÍAS UTILIZADAS

3.1 Proceso de selección

A la hora de elegir las tecnologías que se iban a utilizar para llevar a cabo la aplicación, se debían tener en cuenta varios factores:

- **Conocimientos previos:** antes de comenzar a desarrollar la aplicación, solo había cursado una asignatura de programación, más concretamente sobre el lenguaje de programación C, con lo cual ya tenía los conocimientos básicos a nivel de lógica. En mis ratos libres, ya que me interesaba el tema, había realizado algún curso de Python online y había cambiado algo de código HTML a través de la consola de desarrolladores integrada en los navegadores.
- **Tiempo disponible:** Muy limitado. En consecuencia, no podía comprometerme con ninguna tecnología con una curva de aprendizaje elevada.
- **Recursos online:** Se consideran más accesibles las tecnologías que tuviesen suficiente documentación en internet, y una comunidad activa donde interactuar para la rápida resolución de dudas y problemas.
- **Requerimientos de la aplicación:** las funcionalidades de la aplicación son sencillas y no requerían de nada especial a nivel de dispositivo o APIs.

Teniendo estos factores en cuenta, se decidieron utilizar las siguientes tecnologías, las cuales serán explicadas más adelante en detalle.

Lenguajes	Librerías	Otros
HTML5	AngularJS	Cordova Apache
JavaScript	Bootstrap	PHP Admin
CSS	jQuery	MariaDB
SQL	Font Awesome	Bower
C#	SurveyJS	Visual Studio
		Brackets
		dbdiagram.io

Tabla 1: Tecnologías utilizadas

También hubo una serie de tecnologías que fueron descartadas por varias razones, entre las que destacan:

- **NodeJS**
 - Descripción: Es una librería y entorno de JavaScript cuya función es ejecutar un gran número de funciones y operaciones de manera simultánea, sobretodo centrándose en la parte del servidor.
 - Razón de descarte: Para los requerimientos de la aplicación, y teniendo en cuenta la curva de aprendizaje se prefirió usar peticiones Ajax de *jQuery* para la comunicación con el servidor por su sencillez y ya que no se prevé un volumen alto de peticiones simultáneas al servidor.
- **Onsen UI**
 - Descripción: Es una framework y librería de componentes basados en CSS, HTML5 y JavaScript con el objetivo de desarrollar aplicaciones móviles con aspecto nativo tanto para Android como para iOS, con la necesidad de programar solo una vez dicho elemento para ambas plataformas.

- Razón de descarte: No esta masivamente adoptada, y si bien la documentación era muy completa, en el caso de tener cualquier duda, por ejemplo, los foros destinados a la misma contaban con pocas personas y poca actividad, por lo tanto, si algún problema relevante ocurriese durante el desarrollo podría causar serios retrasos.
- **ionic:**
 - Descripción: es un framework de desarrollo destinado al desarrollo de aplicaciones multiplataforma basado en tecnologías web.
 - Razón de descarte: si bien es cierto que es un framework más completo que Cordova, por esta razón es más complejo, además, por razones que se comentarán más adelante, Cordova fue elegido para realizar esta función fundamental en el desarrollo.

3.2 Proceso de aprendizaje

Tal y como se ha explicitado anteriormente, al comienzo de este trabajo no conocía prácticamente ninguna de las tecnologías a utilizar, pero hay que tener presente que la razón por la que fueron elegidas estas tecnologías fue por su mayor facilidad en la curva de aprendizaje, y la gran comunidad de usuarios y de conocimiento detrás de las mismas.

Ahora vamos a pasar a hablar de las diferentes herramientas y técnicas utilizadas para adquirir los conocimientos necesarios para el desarrollo de la aplicación.

Cursos Online

Existen multitud de plataformas de cursos online tanto de programación como de cualquier temática relacionada con las nuevas tecnologías. Algunas son de pago, otras gratuitas y otras tiene un modelo *fremium*, donde puedes acceder a cursos básicos de manera gratuita, pero por cursos más avanzados o funcionalidades extras requieren un pago.

Las plataformas principales que he usado para obtener los conocimientos necesarios son:

- **PluralSight:** plataforma de pago a la cual tuve acceso por motivo de mi entorno laboral, con cursos sobre todo lo relacionado con programación, muy bien estructurados, y con profesores expertos en dichos temas. Aquí realice los siguientes cursos:
 - HTML, CSS, and JavaScript: The Big Picture (1'5h)
 - JavaScript: Getting Started (3h)
 - JavaScript Fundamentals (3h)
 - Building Mobile Apps with Visual Studio Tools for Apache Cordova (4h)
 - AngularJS: Get Started (3h)
 - AngularJS Fundamentals (7h)
 - Building a Web App with ASP.NET Core, MVC, Entity Framework Core, Bootstrap, and Angular (10h)
 - Node.js: Getting Started (4h)
 - jQuery Fundamentals (5h)
- **Udemy:** plataforma de modalidad *fremium*, donde he realizado el curso "Learn to Build Your First Apache Cordova / PhoneGap App" (4h)

Otras plataformas como Udacity fueron utilizadas, pero para temas puntuales.

Tutoriales Online

Existen multitud de plataformas con tutoriales, tanto dentro de pequeñas webs dedicadas a la programación donde un usuario ha escrito un *post* explicando cómo implementar cierto elemento o plataformas de video como YouTube en la cual hay una infinidad de tutoriales sobre todos los lenguajes, librerías, frameworks, etc...

Documentación oficial

Prácticamente todos los lenguajes, librerías y frameworks cuentan con una documentación donde se explica el funcionamiento de las mismas, ejemplos e incluso tutoriales completos, y las herramientas utilizadas, al ser muy populares cuentan con documentación muy completa.

Foros de preguntas

El más conocido en el sector de la programación es sin ninguna duda Stack Overflow, y allí encontraba solución a cientos de dudas y problemas que surgían durante el desarrollo, en la mayoría de los casos, la duda y solución ya existían, pero en contadas ocasiones ha sido necesario formular yo las preguntas de manera concisa para obtener la solución apropiada.

Otros foros de preguntas han sido utilizados con seguridad, ya que, al buscar las preguntas en google, uno tiende a elegir la primera opción que le aparece, muchos de estos foros solían ser específicos de la tecnología relativa a la duda o error.

Buenos hábitos

Viendo los tutoriales no se aprende a programar, los tutoriales generalmente animan al espectador a practicar programando mientras uno sigue el curso. Además, es muy recomendable hacer apuntes donde poder acudir cuando estés programando para recordar cómo se realizaba la función que quieres implementar.

Otra herramienta muy útil para probar cosas sobre la marcha, hacer pequeños experimentos antes de implementarlos, es utilizar *Coding Playgrounds* como son JSFiddle, CodePen o .Net Fiddle, por nombrar algunos. En estas páginas web, uno puede escribir pequeños programas y probarlos de manera rápida y cómoda, teniendo la posibilidad de guardarlos para futura utilización.

3.3 Apache Cordova

Esta es la tecnología base del proyecto, es un marco de desarrollo de código abierto destinado a plataformas móviles. Utilizando lenguajes de desarrollo web como son HTML5, CSS3 y JavaScript, permite desarrollar para varias plataformas móviles de manera simultánea sin la necesidad de utilizar los lenguajes nativos de cada plataforma. También existen múltiples *plugins* mediante los cuales uno puede acceder a funcionalidades propias del teléfono, como son la cámara, GPS o sensores.

Sus orígenes se remontan al año 2008, donde nació PhoneGap de la mano de una compañía denominada Nitobi Software. En el año 2011 fue adquirida por Adobe Systems y fue distribuida como código libre pasando a llamarse Apache Cordova, ya que fue donada a la Apache Software Foundation, que es una corporación sin ánimo de lucro cuyo objetivo es distribuir herramientas de código libre con el objetivo de mejorar el internet.

El nombre que se le dan a las aplicaciones desarrollados con este tipo de marcos de desarrollo es “aplicaciones híbridas”, ya que incorporan el desarrollo web con funcionalidades específicas de cada plataforma, reduciendo el coste y tiempo de desarrollo respecto a las aplicaciones nativas. El principal inconveniente de estas es la falta de optimización respecto a la fluidez, pero gracias al avance computacional de los dispositivos móviles, esto ya no es un problema relevante y los beneficios superan con creces dicho inconveniente.

3.4 Entorno de desarrollo

Visual Studio

El entorno de desarrollo principal donde vamos a desarrollar la aplicación es Microsoft Visual Studio, por una gran de razones.

La primera razón sería que es gratuito en su Community Edition, la versión utilizada, más concretamente ha sido *Microsoft Visual Studio Community Edition 2017*.

Cuenta con un gran ecosistema de extensiones y servicios integrados que facilitan mucho a la hora de programar y también más concretamente para el desarrollo de proyectos basados en Cordova Apache, las más relevantes de las utilizadas:

- Visual Studio Tools para Apache Cordova: permite desarrollar aplicaciones para este marco de desarrolla de manera cómoda gracias a la creación de proyecto con los ficheros necesarios para empezar, emulador y vista previa integrada, sistema de gestión de paquetes (librerías y extensiones), y empaquetamiento del proyecto en las aplicaciones correspondientes.
- IntelliSense: herramienta implementada cuya funcionalidad es la de completar código, sugerir cambios, y ordenar la estructura del código.
- Herramientas de depuración y análisis: permitiendo revisar el código paso por paso, colocando interrupciones en los puntos problemáticos con el objetivo de analizar en detalle lo que ocurre a nivel de código y variables.
- Control de versiones: es una extensión que se puede descargar y se hablará de la misma más adelante.

Brackets

Es un entorno de desarrollo mucho más simple y básico, más concretamente es un editor de código. Fue utilizado para desarrollar las *queries* de SQL para la creación y población de la base de datos, ya que carecía de la extensión de Visual Studio compatible con este lenguaje.

3.5 Lenguajes de Programación

Han sido utilizados múltiples lenguajes y a continuación se pasará a explicar porque se ha elegido cada uno y sus propiedades.

HTML

Su nombre son las siglas de *HyperText Markup Language* (lenguaje de marcas de hipertexto), su utilidad es la elaboración de páginas web y es un estándar a cargo del *World Wide Web Consortium*. Define la estructura de los elementos que conforman la página web.

Los navegadores son los encargados de interpretar este código y representarlo según los estándares al usuario a través del mismo. Es un estándar que evoluciona, pero los navegadores tienen retrocompatibilidad con el objetivo de no dejar obsoletas páginas web que no han sido actualizadas a estándares modernos.

CSS

Estas son las siglas de *Cascading Style Sheets* (hojas de estilos en cascada), el cual es un lenguaje de diseño gráfico cuyo objetivo es establecer el diseño y aspecto que tendrán los diferentes elementos que conforman el código HTML que contiene la información de la página web.

Las principales características que se establecen son los colores, fuentes, capas o *layouts*, una gran ventaja es que varios elementos pueden compartir una definición de diseño, y al cambiar esta, se cambia para todos de manera sencilla y eficiente.

Tiene una organización jerárquica donde las categorizaciones más concretas tienen una mayor prioridad a la hora de definir las características del elemento, con el objetivo de hacer más flexible su utilización, ya que un mismo elemento puede tener varias descripciones que se contradicen y alguna debe tener prioridad sobre la otra.

JavaScript

Es un lenguaje de programación interpretado, ya que no requiere compilación previa, en el caso de páginas web, es el navegador web el que lo ejecuta conforme es necesario. Al igual que el HTML es un estándar, y en este caso, a cargo de ECMA (*European Computer Manufacturers Association*), quienes lo actualizan periódicamente añadiendo nuevas funcionalidades y solucionando problemas.

Su papel es el de añadir dinamismo a las páginas web, ya que previamente, estas eran estáticas, meros visualizadores de información. Permite manipular la estructura, responder a eventos, realizar cálculos en el momento y muchas más funciones. Se encarga de toda la lógica de la aplicación y existen infinidad de librerías que añaden funcionalidades y facilitan el trabajo en múltiples campos de programación, estas se pueden descargar o enlazar para que se descarguen dentro del navegador cuando haga falta.

Las funciones son lo más interesante de JavaScript, estas permiten realizar una tarea o cálculo. En nuestro caso, estas funciones son definidas en el controlador y pueden ser llamadas desde la vista directamente o desde el propio controlador al reaccionar a algún evento producido.

JSON

Son las siglas de *JavaScript Object Notation*, el cual es un formato de archivo de estándar abierto que se usa para la transmisión de datos. Originariamente su utilidad era para representar objetos de JavaScript de una forma fácilmente interpretable, pero ha ido ganando popularidad y se utiliza en múltiples lenguajes como por ejemplo *.NET* el cual exporta clases de este lenguaje o para poblar código XML.

SQL

Por sus siglas *Structures Query Language* (lenguaje de consulta estructurada), es el programa utilizado para programar, administrar y utilizar sistemas de gestión de base de datos.

Este lenguaje ha sido utilizado para la creación y diseño de la base de datos necesaria para almacenar información de la aplicación y del hospital para ofrecer las funcionalidades necesarias. Otro uso, ha sido el de poblar las diferentes tablas de la base de datos con datos de prueba para poder programar y comprobar el correcto funcionamiento de la aplicación, y para ello se ha debido programar las ordenes con la información que se deseaba añadir a la base de datos.

C#

También conocido como *C Sharp*, es un lenguaje de programación orientado a objetos desarrollado por Microsoft y deriva de otros lenguajes como C/C++.

No se entrará en detalle, ya que solo ha sido utilizado para la programación “automática” de las *queries* de SQL de población de datos para la base de datos, simplemente por era el lenguaje con el que me sentía más cómodo a la hora de manipular *strings*, bucles y aleatorización de variables.

3.6 Librerías

Pasamos a explicar brevemente las librerías principales utilizadas y su funcionalidad dentro de la aplicación.

Estas librerías han sido obtenidas por medio del gestor de paquetes Bower, el cual nos permite descargar librerías, actualizarlas y gestionar todo lo relacionado con las mismas.

AngularJS

Al igual que Cordova es la base de nuestro proyecto, AngularJS será la base de nuestra aplicación web. Este marco de desarrollo basado en JavaScript, mantenido por Google, se encargará de crear y hacer funcionar nuestra aplicación en una sola página, simplificando el desarrollo.

Más concretamente de encarga de permitirnos utilizar el patrón de arquitectura MVC (modelo vista controlador), en el cual el HTML y CSS formarán la vista, lo que ve el usuario, y el JavaScript hará de controlador, aportando lógica a la página correspondiente, pero todo bien organizado, cada vista con su controlador y ruta de acceso.

Bootstrap

Es una biblioteca o conjunto de herramientas de código abierto para diseño de páginas o aplicaciones web. Contiene elementos y directrices basadas en HTML, CSS y JavaScript, para poder implementar de manera sencilla y elegante en nuestra web.

jQuery

Es una biblioteca basada en JavaScript, simplifica la manera de interactuar con los elementos del código, controlar eventos, desarrollar animaciones y agregar comunicación con base de datos mediante AJAX, que será lo que utilizaremos. Una de las mayores ventajas es el hecho de que sea multiplataforma, y que funcione en todos los navegadores por igual a pesar de las peculiaridades de cada uno, simplificando el código enormemente.

El control de eventos ha sido ampliamente utilizado en el desarrollo ya que es superior al nativo de JavaScript tanto por sencillez como funcionalidad.

La comunicación mediante AJAX, se explicará en detalle más adelante, ya que es una parte fundamental de la aplicación para ser funcional, pero simplemente consiste en el envío y recepción de información por medio de JSON, explicado anteriormente.

Font Awesome

Es una biblioteca de iconos, de fácil implementación en nuestro proyecto y que se adapta al texto con apariencia natural. Al ser imágenes vectoriales, escalan a cualquier tamaño sin perder calidad y al tratar dichos iconos como fuente son totalmente personalizables mediante reglas de estilo.

SurveyJS

Consiste en una librería de JavaScript que permite crear y customizar de manera sencilla encuestas y formularios. El principal valor que aporta es la flexibilidad, ya que permite guardar las encuestas como un JSON (explicado anteriormente), y las respuestas también, facilitando el almacenamiento del mismo y envío de respuestas. Es muy personalizable, desde quien realizara la interpretación (en nuestro caso jQuery), el tema y muchas más opciones de diseño.

3.7 Control de versiones

Este ha sido llevado a cabo por la extensión de GitHub para Visual Studio, por tenerlo integrado en el propio entorno de desarrollo. Dentro de mismo entorno de desarrollo uno puede sincronizar, bifurcar, recuperar y extraer código del repositorio.

GitHub es un servicio de almacenaje de control de versiones usando Git, un sistema de seguimiento de cambios en el código destinado a coordinar el trabajo de varios programadores en un mismo proyecto y asegurar la integridad del código.

3.8 Base de Datos

La tecnología utilizada para la base de datos se divide en tres elementos, herramienta de diseño (dbdiagram.io), la herramienta de administración de la base de datos (PhpMyAdmin) y el gestor de base de datos relacionales (MariaDB).

dbdiagram.io

Servicio web que permite el diseño de manera sencilla y visual de las tablas de una base de datos, las relaciones entre las mismas y propiedades de las columnas.

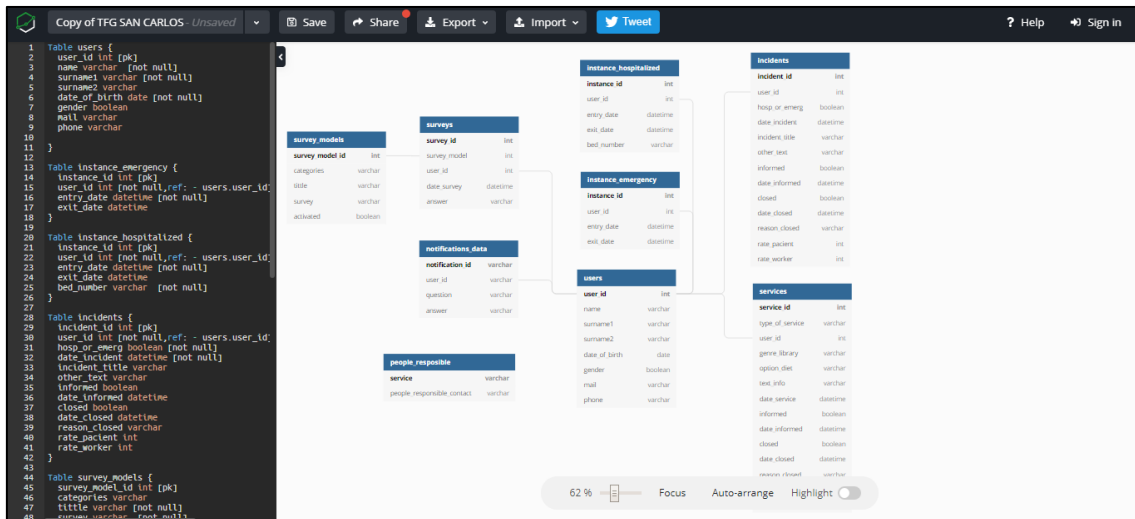


Ilustración 1: Herramienta dbdiagram.io

En el panel de la izquierda se programa el diseño con un lenguaje propio que carece de dificultad y se entiende con una simple lectura. A su derecha tenemos la representación de la base de datos, lo cual es muy útil para tener una visión de los distintos elementos y como están relacionados.

Permite en la parte superior exportar e importar el diseño, lo cual es tremendamente útil, ya que desarrolla el código de creación de la base de datos automáticamente, sin embargo, por falta de algunas funcionalidades, ese código debía ser completado para poder incluir todas las características necesarias.

PhpMyAdmin

Es un administrador web portable de MySQL y MariaDB, gratuito y de código abierto, permite acceder a la base de datos de internet de manera sencilla, realizar las operaciones deseadas con la misma, exportar e importar datos, y muchas otras funcionalidades.

MariaDB

Es un gestor de base de datos relacionales basada en MySQL, es gratuito y de código abierto. Es el software que permite al usuario definir, crear, mantener y controlar el acceso a la base de datos.

CAPÍTULO 4: DISEÑO Y FUNCIONALIDADES DE LA APLICACIÓN

4.1 Proceso de Diseño

Todo comienza con una reunión con el cliente para conocer el proyecto, funcionalidades principales, y detalles básicos de la aplicación.

Una vez que tenemos esta información podemos pasar a desarrollar el Mock-up, es un prototipo a nivel de diseño de la aplicación, consta de varios pasos:

- **Diagrama de flujo del usuario:** Donde especificamos los pasos que va a seguir el usuario dentro de nuestra aplicación para realizar las diferentes acciones, para tener una idea conceptual de las páginas y elementos necesarios.
- **Wireframe:** Es un prototipo visual de la estructura básica de la aplicación, realizado con formas y elementos simples.
- **Patrón de diseño y paleta de colores:** en esta etapa se eligen los diferentes elementos principales de la aplicación, fuentes y colores;
- **Mock-up:** es el resultado de todo el proceso, donde tenemos un prototipo a nivel de diseño de la aplicación y la relación entre las diferentes páginas que conforman nuestra aplicación.

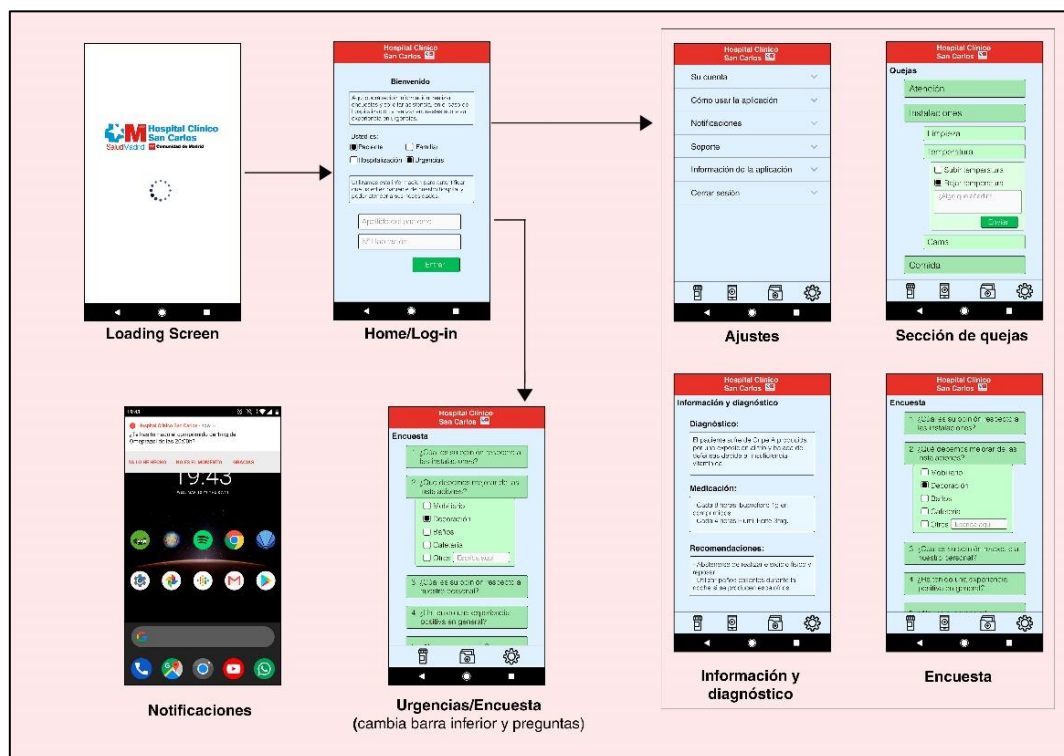


Ilustración 2: Mock-up preliminar de la aplicación

Este es el diseño inicial para comenzar a programar, la base, pero conforme el proyecto toma forma y se realizan más reuniones con el cliente, se van perfilando detalles y añadiendo o quitando funcionalidades según se vea necesario.

La paleta de colores final cambió, habiendo sido utilizados los siguientes colores:

Uso	Color (HEX)
Barra de notificaciones	#fc7373
Bonotes de envío	#43ca17
Tarjetas de información	#1e90ff
Tarjetas secundarias	#cef8ff
Color texto botones	#009688

Tabla 2: Colores utilizados

4.2 Funcionalidades

El prototipo de la aplicación de cara a los pacientes tendrá disponible las siguientes funcionalidades que se especifican a continuación.

Una vez descargada la aplicación, al entrar por primera vez, aparecerá una pantalla de acceso donde el usuario dependiendo de su perfil (urgencias o hospitalizado) podrá autenticarse basándose en los datos que tiene el hospital de cada paciente, para poder verificar que es la persona en cuestión (o un familiar) el que está haciendo uso de la aplicación.

Al acceder, es redireccionado a una pantalla principal desde la cual podrá acceder a los principales apartados de la aplicación (encuestas, incidencias, servicios y ajustes), pasamos a explicar cada una de ellas.

En la sección de encuestas, el usuario podrá seleccionar entre las encuestas que se encuentran disponibles para su categoría de usuario y proceder a la cumplimentación de las mismas.

En el apartado de incidencias, se le ofrece al usuario un menú predeterminado de desplegables a través del cual podrá seleccionar la tipología de incidencia afectada, y en el caso de que no estuviese en ninguna de las opciones predeterminadas, podrá redactar la incidencia en una sencilla y convencional caja de texto. Además, habrá una pestaña para acceder al seguimiento de incidencias del usuario.

Los servicios se solicitan desde una pantalla en la cual el usuario selecciona el servicio deseado, algunos servicios como biblioteca o dieta, tienen opciones específicas para su elección, y a parte el usuario tiene la posibilidad de escribir en un cuadro de texto información suplementaria de lo que necesita. Al igual que en el caso de las incidencias, también hay un botón para acceder al seguimiento del servicio solicitados.

En los ajustes, el usuario podrá acceder a información básica de su perfil, cerrar sesión, acceder a información de la aplicación y controlar las notificaciones.

4.3 Diagrama de casos de uso

Un diagrama de casos de uso es un diagrama dinámico en UML (Unified Modeling Language), modela la funcionalidad de un sistema usando actores y casos de uso. Siendo casos de uso una serie de acciones, servicios y funciones que el sistema necesita desarrollar. En este contexto, un sistema es algo que está siendo desarrollado u operado como por ejemplo una página web o aplicación (como es este caso). Los actores son personas o entidades operando bajo roles definidos dentro del sistema.

Pasamos a analizar los distintos casos de uno de la aplicación, con los agentes implicados en los mismos y las relaciones entre estos.

El agente primario de la aplicación es el usuario, es decir el paciente del hospital, ya que la mayoría de las funciones están centradas en ofrecerle un mejor servicio y es el que activamente utiliza la aplicación en sí.

El otro agente es el hospital, formado por las personas responsables de solucionar las incidencias, de ofrecer los servicios y de analizar los resultados de las encuestas.

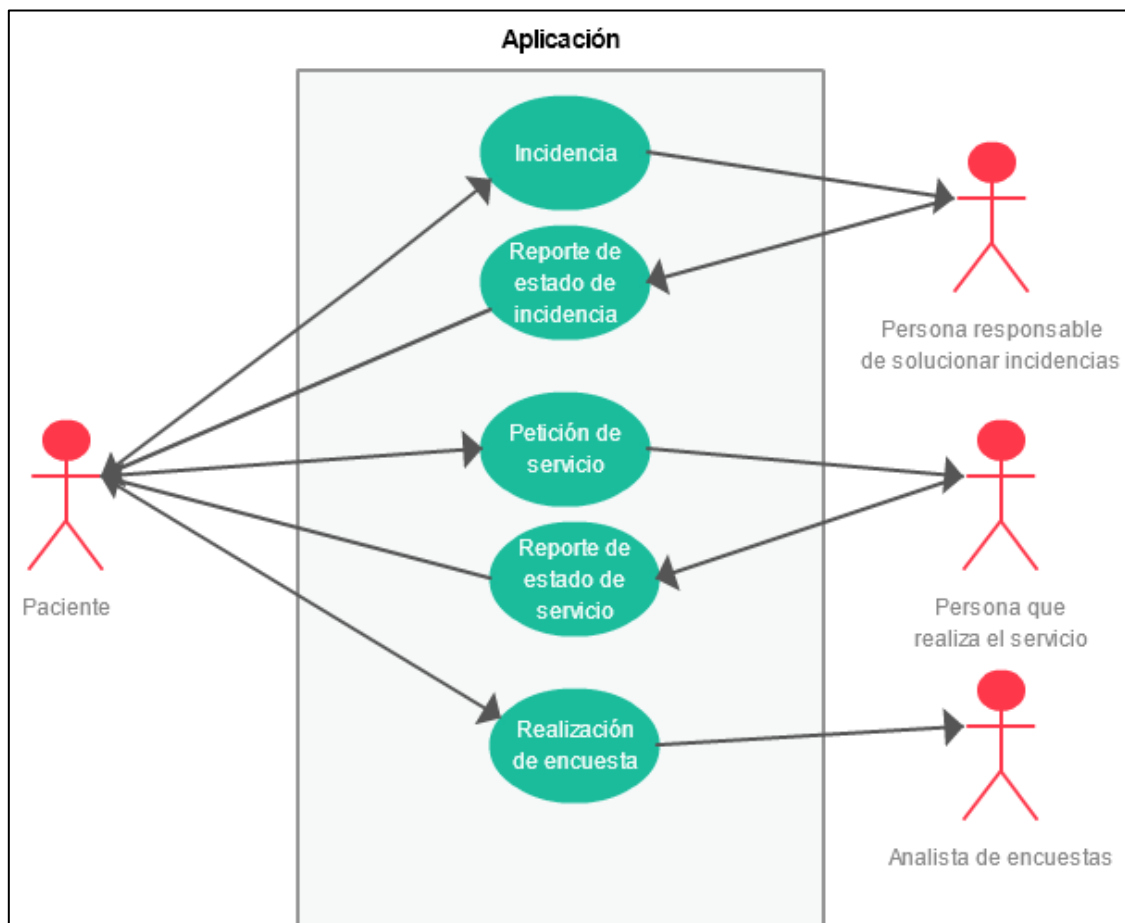


Ilustración 3: Casos de uso

4.4 Modelo de la aplicación

En este apartado, se muestra la estructura de la aplicación que permite al usuario realizar las diferentes acciones dentro de la aplicación. El objetivo de esta estructura es facilitar al usuario el uso de la aplicación, pudiendo realizar las acciones de manera intuitiva y rápida.

El inicio de la aplicación es la página de acceso, en el caso de no haber accedido previamente, y en caso de haber realizado este paso con anterioridad, aparecerá en el menú principal directamente, donde podrá acceder a las diferentes secciones de la aplicación.

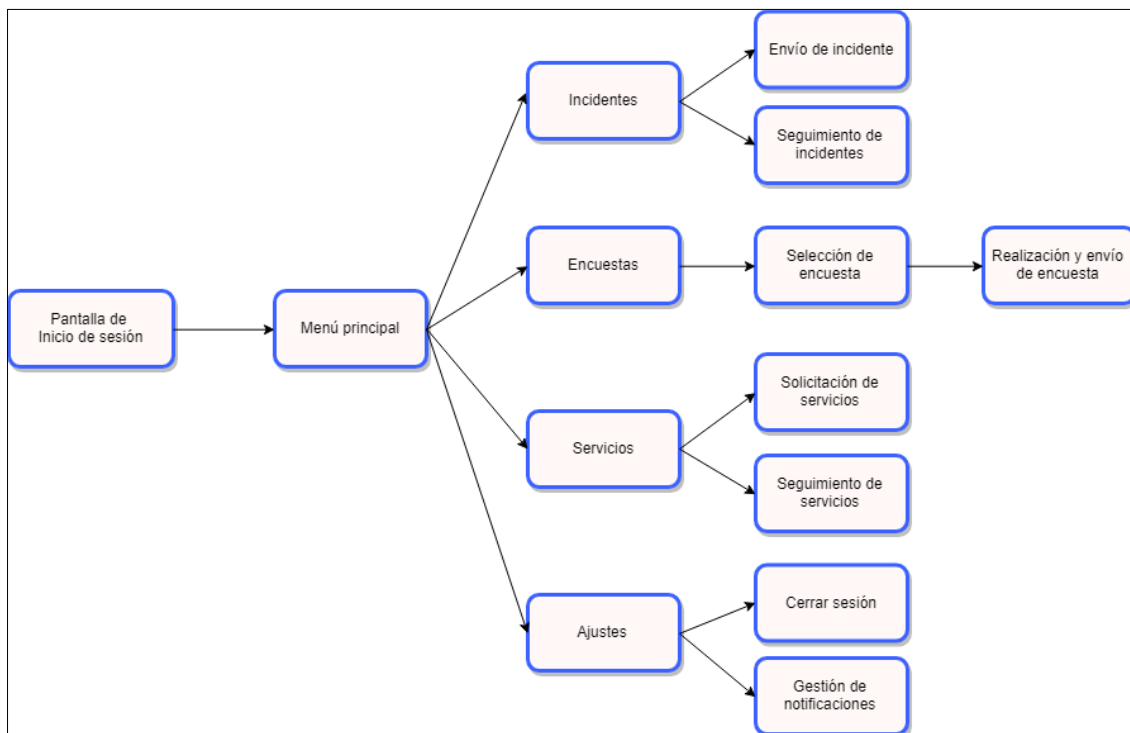


Ilustración 4: Modelo de la aplicación

4.5 Diseño de pantallas

El diseño de las pantallas ha intentado ofrecer la mejor experiencia al usuario, centrándose en hacer la aplicación intuitiva, ya que muchos de los potenciales usuarios pueden ser que, por edad, u otras razones no estén inmersos en el mundo de las nuevas tecnologías, como las generaciones más jóvenes. Por esta razón se han elegido imágenes para acompañar a o ser los enlaces y aportar una experiencia más visual e intuitiva.

A continuación, se muestra una tabla con las diferentes imágenes utilizadas.





Nombre	Imagen
Servicios.png	
Incidencias.png	
Encuestas.png	
Ajustes.png	

Tabla 3: Imágenes utilizadas

Estos son los iconos utilizados de la librería Font Awesome:











Nombre	Imagen	Nombre	Imagen
fa-clipboard-list		fa-utensil-spoon	
fa-book		fa-user	
fa-calendar-week		fa-ambulance	
fa-building		fa-id-card	
fa-utensils		fa-venus-mars	

Tabla 4: Iconos utilizados

CAPÍTULO 5: DESARROLLO DE LA APLICACIÓN

5.1 Páginas

En este apartado se explicará el funcionamiento de las diferentes páginas que componen la aplicación al mismo tiempo que se muestran los diagramas de flujo de las mismas.

5.1.1 Página Index

Esta es la página donde ocurre todo, ya que gracias a la librería AngularJS, en verdad es una página web de una sola página, pero va cargando las diferentes páginas en un elemento en la página *index*, simplificando la inclusión de la barra de navegación, la cual se encuentra en la página superior. El elemento utilizado para introducir las diferentes paginas es `<ng-view>`, como se observa a continuación:

```
<body>
<div style="position: relative;">
  <!--Barra de navegación-->
  <nav id="Barra Menu" class="navbar navbar-expand-lg navbar-light bg-light" style="z-index:10; position:absolute; background-color:#fc7373 !important; width:100%">
    <a class="navbar-brand" href="#">Hospital San Carlos</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation" >
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNav">
      <ul class="navbar-nav">
        <li class="nav-item">
          <a class="nav-link" style="color:black" data-toggle="collapse" data-target=".navbar-collapse.show" href="index.html#!/main">Main</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" style="color:black" data-toggle="collapse" data-target=".navbar-collapse.show" href="index.html#!/quejas">Incidencias</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" style="color:black" data-toggle="collapse" data-target=".navbar-collapse.show" href="index.html#!/selectencuesta">Encuestas</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" style="color:black" data-toggle="collapse" data-target=".navbar-collapse.show" href="index.html#!/servicios">Servicios</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" style="color:black" data-toggle="collapse" data-target=".navbar-collapse.show" href="index.html#!/ajustes">Ajustes</a>
        </li>
      </ul>
    </div>
  </nav>
  <!--Espacio detrás de la barra-->
  <div style="width:1px; height:50px;"></div>
  <!--Donde se introduce la vista de cada página-->
  <div ng-view ></div>
</div>
</body>
```

Ilustración 5: Código HTML barra de navegación

En esta página también se cargan las diferentes librerías, páginas de estilos y controladores en la cabecera.

5.1.2 Inicio de sesión

Accediendo por primera vez a la aplicación o al haber cerrado la sesión anteriormente, el usuario debe introducir datos que lo autentifiquen y que demuestren que se encuentra en el hospital y que es quien dice ser. También se puede aceptar la política de privacidad. Aquí el diagrama de flujo:

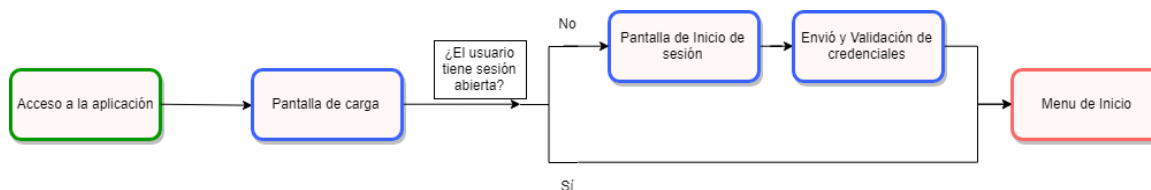


Ilustración 6: Diagrama de sección Inicio de sesión

Para comprobar si el usuario ha iniciado la sesión y redirigirlo hacia el menú principal se realiza mediante la siguiente función, que comprueba el valor correspondiente almacenado en memoria.

```
//Si ya esta logeado lo manda a Main
$(document).ready(function () {
    if (localStorage.getItem('signedIn') == 'Yes') { window.location.href = "index.html#!/main"; }
});
```

Ilustración 7: Código JavaScript redirección sesión iniciada

Para poder enviar las credenciales, estas tienen que cumplir con unos requisitos, como por ejemplo que estén todas rellenas. Esta función de validar es llamada en el controlador cada vez que se rellena algún campo, gracias a un evento de JavaScript, y cuando sea válido el formulario, se puede pulsar el botón de enviar.

```
//Valida el formulario
$scope.validateForm = function () {
    var Area = $("#Area").val();
    var Perfil = $("#Perfil").val();
    var Apellido = $("#Apellido").val();
    var Hab = $("#Hab").val();
    var Genero = $("#Genero").val();
    var FechaNac = $("#FechaNac").val();
    if (Area == "Area") { return false }
    if (Perfil == "Perfil") { return false }
    if (Apellido == "") { return false }
    if (FechaNac == "") { return false }
    if (Genero == "" && Hab == "") { return false }
    if ($scope.hasNumber(Apellido)) { return false }
    else return true
}
```

Ilustración 8: Código JavaScript validación formulario

Esa información es enviada a través de una llamada AJAX al servidor, el cual nos devuelve el id del usuario (*user_id*) en caso de que este todo correcto y en caso contrario, un código de error correspondiente con el error. A continuación, se ve la correspondiente a un usuario hospitalizado, pero la correspondiente a un usuario de urgencias es similar.

```
function loginAjax(Area, Apellido, Hab, Genero, fechaNac, Perfil) {
    if (Area=="HOSPITALIZADO") {
        var datos = {"area": Area,"apellido":Apellido,"fechanac":fechaNac,"habitacion":Hab};
        $.ajax({
            url: "http://138.100.72.88/hsc/usuarios/checklogin.php",
            type: "POST",
            dataType: 'json',
            data: JSON.stringify(datos),
            processData: false,
            success: function (data) {
                if (data.status == 'KO') {
                    alert("Los datos introducidos NO son correctos");
                }
                if (data.status == 'OK') {
                    console.log(data);
                    localStorage.setItem('user_id', data.data[0].user_id);
                    return true;
                }
                else {
                    console.log(data);
                }
            }
        });
    }
}
```

Ilustración 9: Código AJAX Inicio sesión

5.1.3 Menú inicial

Se accede a esta página una vez que el usuario está registrado, en esta página, el usuario podrá elegir a que otra página acceder, incidencias, encuestas, servicios o ajustes. Aquí está el diagrama de flujo:

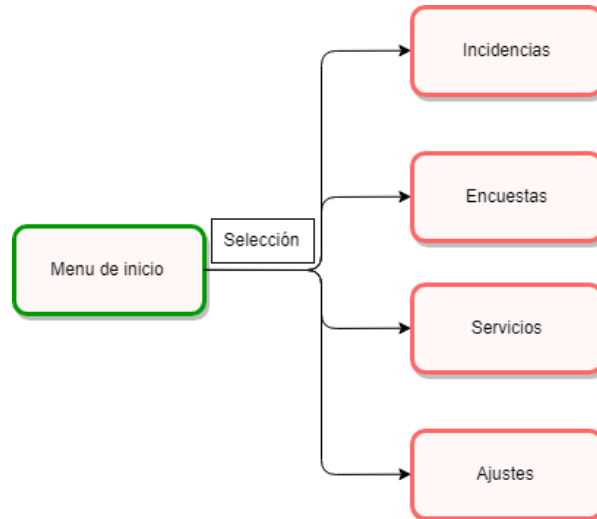


Ilustración 10: Diagrama de sección Menú de inicio

En el caso de que usuario no haya iniciado sesión será redirigido a la página de inicio de sesión con un código similar al mostrado en el apartado anterior.

Al pulsar cualquier imagen, se llama a la función *redirect*, la cual redirecciona a la página correspondiente:

```
//Funcion para redireccionar con los botones
$scope.redirect = function (index) {
  if (index == '1') { window.location.href = "index.html#!/quejas"; }
  else if (index == '2') { window.location.href = "index.html#!/encuesta"; }
  else if (index == '3') { window.location.href = "index.html#!/ajustes"; }
  else if (index == '4') { window.location.href = "index.html#!/servicios"; }
}
```

Ilustración 11: Código JavaScript redirección menú de inicio

5.1.4 Incidencias

En el siguiente diagrama de flujo podemos ver de manera visual las diferentes opciones dentro de las páginas:

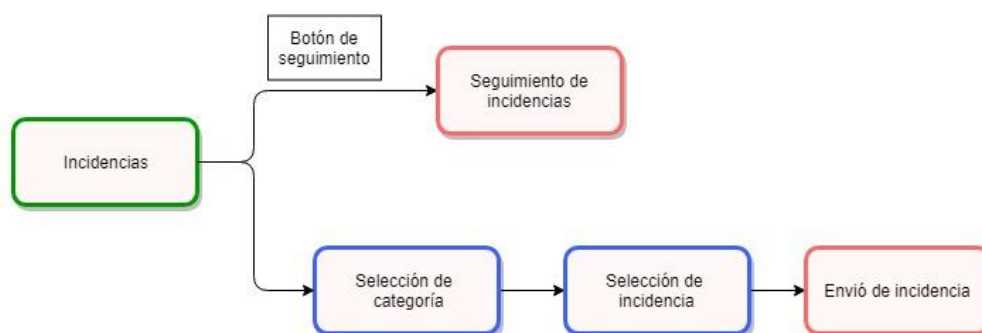


Ilustración 12: Diagrama de sección Incidencias

Aplicación móvil para recabar la opinión de los pacientes sobre la calidad del servicio sanitario percibido

Para generar las diferentes opciones y hacerlo más flexible, se han creado las diferentes opciones mediante un bucle `<ng-repeat>` perteneciente a AngularJS que recorre unas variables creadas basándose en dos clases, *bloque* y *pregunta*, los bloques almacenan preguntas, y cada pregunta tiene diferentes opciones.

```
//Clase Pregunta
var pregunta = function (titulo, id, ...opciones) {
  this.titulo = titulo;
  this.opciones = opciones;
  this.id = id;
}

//Clase Bloque (de preguntas)
var bloque = function (index, supertitulo, ...preguntas) {
  this.supertitulo = supertitulo;
  this.preguntas = preguntas;
  this.index = index;
}
```

Ilustración 13: Código de clases incidencias

```
<!--Formulario en si, que se crea con bucles-->
<form>
  <div ng-repeat="bloque in elementos track by $index">
    <!--Titulo de cada bloque (pinchando en el se cierran los demás bloques)-->
    <div ng-click="togglepreguntas($index,bloque.index)">
      <div class="card text-white bg-info mb-3" style="position: absolute center; margin-left: 3%; margin-right: 3%; margin-top: 10px; background-color: dodgerblue !important; color: white;">
        <div class="card-header text-center">{{bloque.supertitulo}}</div>
      </div>
    </div>
    <div class="preguntas[{{bloque.index}}][{{index}}]" style="display: none">
      <!--Preguntas de cada bloque-->
      <div ng-repeat="pregunta in bloque.preguntas track by $index">
        <div class="card border-primary" style="margin-left: 5%; margin-right: 5%; margin-top: 5px; ng-click="toggleopciones($index, bloque.index)">
          <div class="card-body text-primary" style="background-color: #e8f5e9">
            <p class="card-text" style="font-size: small; margin: -15px; justify-content: flex-end; padding: 10px; color: black">{{pregunta.titulo}}</p>
          </div>
          <!-- Opciones de cada pregunta -->
          <div class="opciones[{{bloque.index}}][{{index}}]" style="display: none; margin-left: 7%; margin-right: 7%; background-color: #d9f9d9; text-align: center;">
            <div class="form-group input-group">
              <div class="input-group-prepend">
                <span class="input-group-text"><i class="fa fa-question fa-xs" style="margin-right: 10px; margin-left: 10px"></i></span>
              </div>
              <select id="select[{{bloque.index}}][{{index}}]" class="form-control" ng-model="pregunta.id">
                <option ng-selected="1" value="" hidden>Seleccione una opción</option>
                <option ng-repeat="opcion in pregunta.opciones" value="{{opcion}}">{{opcion}}</option>
              </select>
            </div>
            <!-- Si se selecciona la opcion otro aparece un campo para escribir -->
            <div ng-if="pregunta.id == 'Otro'" style="width: 70%; margin: 0 auto">
              <textarea style="display: inline-block;" id="textOtro[{{bloque.index}}][{{index}}]" rows="4" cols="30" required placeholder="Especifique aqui su queja.">
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</form>
```

Ilustración 14: Código HTML incidencias

En esta página el usuario podrá informar de cualquier incidencia producida, tanto para usuarios hospitalizados como de urgencias, según esta distinción, tendrá opciones diferentes. Para esto se ha utilizado la siguiente el siguiente código:

```
if (localStorage.getItem('Area') == 'Hospitalizado') {
  //En esta variable de almacenan todos los bloques y preguntas de hospitalizados
  $scope.elementos = [bloque1, bloque2, bloque3, bloque4];
}
if (localStorage.getItem('Area') == 'Urgencias') {
  //En esta variable de almacenan todos los bloques y preguntas de urgencias
  $scope.elementos = [bloque11, bloque22, bloque44];
}
```

Ilustración 15: Código opciones incidencias

Para realizar la animación de minimizar bloques y opciones, se ha hecho a mano usando la lógica y siguiendo los índices de los bucles necesarios para crear la página:

```
//Esta función se encarga de hacer desaparecer y aparecer las PREGUNTAS
$scope.togglepreguntas = function (index, bloqueindex) {
    var nombreclase = "preguntas" + bloqueindex + index;
    var x = document.getElementsByClassName(nombreclase);
    for (var i = 0; i < $scope.elementos.length; i++) {
        for (var j = 0; j < 5; j++) {
            var nombreresto = "preguntas" + j + i;
            var y = document.getElementsByClassName(nombreresto);
            try { y[0].style.display = "none"; }
            catch (err) { }
        }
    }

    for (var i = 0, length = x.length; i < length; i++) {
        if (x[i].style.display === "none") {
            x[i].style.display = "block";
        } else {
            x[i].style.display = "none";
        }
    }
}

//Esta función se encarga de hacer desaparecer y aparecer las OPCIONES de las preguntas
$scope.toggleopciones = function (index, bloqueindex) {
    var nombreclase = "opciones" + bloqueindex + index;
    var x = document.getElementsByClassName(nombreclase);
    for (var i = 0; i < 5; i++) {
        var nombreresto = "opciones" + bloqueindex + i;
        var y = document.getElementsByClassName(nombreresto);
        try { y[0].style.display = "none"; }
        catch (err) { }
    }

    for (var i = 0, length = x.length; i < length; i++) {
        if (x[i].style.display === "none") {
            x[i].style.display = "block";
        } else {
            x[i].style.display = "none";
        }
    }
}
```

Ilustración 16: Código JavaScript animaciones incidencias

En caso de necesitar explicar más concretamente o que ninguna opción se adecue, en ambos casos puede escribir con texto libre. El botón de envío no se activará hasta que se haya seleccionado alguna opción, para realizar dicha comprobación se usan las siguientes funciones:

```
//Cuando cambia alguna opción comprueba si puede activar el boton de enviar
$(document).ready(function () {
    $("select").change(function () {
        if ($scope.validateForm()) {
            document.getElementById("#boton-enviar").disabled = false;
            document.getElementById("textTermina").style.display = "none";
        }
    });
});

//Valida que se haya elegido alguna queja
$scope.validateForm = function () {
    var count = 0;
    for (var i = 0; i < $scope.elementos.length + 1; i++) {
        for (var j = 0; j < 5; j++) {
            if ($("#select" + i + j).val() != "") { count = count + 1; }
        }
    }
    if (count > 0) { return true; }
    else { return false }
}
```

Ilustración 17: Código JavaScript de validación incidencias

Al hacer clic en enviar, aparece un *pop-up* de confirmación, el cual es un elemento *modal* de la librería Bootstrap, donde el usuario puede cancelar o enviar la incidencia:

```
<!-- Menu Pop-up de confirmación de envío -->
<div class="modal fade" id="exampleModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <h6 class="modal-title" id="exampleModalLabel">¿Seguro que desea enviar la encuesta?</h6>
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-dismiss="modal">No</button>
        <button type="button" class="btn btn-primary" ng-click="sendService();">Si</button>
      </div>
    </div>
  </div>
</div>
```

Ilustración 18: Código HTML de menú confirmación de envío

En esta página también se accede al seguimiento de incidencias, donde el usuario puede ver en una tabla las incidencias que ha reportado y el estado de las mismas. El proceso de representación ha sido similar al realizado para generar las opciones, mediante un bucle `<ng-repeat>`:

```
<!--Tabla de incidencias-->
<table class="table table-striped">
  <thead>
    <tr>
      <th scope="col" style="font-size:small">Fecha</th>
      <th scope="col" style="font-size:small">Incidencia</th>
      <th scope="col" style="font-size:small">Informado</th>
      <th scope="col" style="font-size:small">Cerrado</th>
    </tr>
  </thead>
  <tbody>
    <tr ng-repeat="item in elementos">
      <td style="font-size:small">{{item.fecha}}</td>
      <td style="font-size:small">{{item.incidencia}}</td>
      <td style="font-size:small">{{item.informed}}</td>
      <td style="font-size:small">{{item.closed}}</td>
    </tr>
  </tbody>
</table>
```

Ilustración 19: Código HTML seguimiento incidencias

El envío de la incidencia se realiza como el resto de comunicaciones con la base de datos, por medio de una llamada Ajax, la cual podemos ver a continuación:

```
function sendincidentAjax(user, hosporeme, titulo, otro) {
  var datos = { "userid": user, "hosporeme": hosporeme, "title": titulo, "other": otro };
  console.log(datos);
  $.ajax({
    url: "http://138.100.72.88/hsc/incidents/sendincident.php",
    type: "POST",
    dataType: 'json',
    data: JSON.stringify(datos),
    processData: false,
    success: function (data) {
      if (data.status == 'KO') {
        alert("Ya se ha enviado correctamente el incidente");
      }
      if (data.status == 'OK') {
        console.log(data);
        return true;
      }
      else {
        console.log(data);
      }
    }
  });
}
```

Ilustración 20: Código AJAX de envío de incidencias

Capítulo 5: Desarrollo de la aplicación

La llamada Ajax para obtener las diferentes incidencias del usuario junto con el estado de las mismas se puede ver a continuación, con estos datos también se crean los diferentes objetos de clase *incidencia* y se juntan en un elemento con el objetivo de compartir dichas incidencias desde el controlador por medio de a la directiva *\$scope*.

```
var incidencia = function (incidencia, fecha, informed, closed) {  
    this.incidencia = incidencia;  
  
    this.fecha = fecha;  
    if (informed == 1) { this.informed = "Sí" }  
    if (informed == 0) { this.informed = "No" }  
    if (informed == null) { this.informed = "No" }  
    if (closed == 1) { this.closed = "Sí" }  
    if (closed == 0) { this.closed = "No" }  
    if (informed == null) { this.closed = "No" }  
}
```

Ilustración 21: Código AJAX de clase incidencia

```
function getIncidencias() {  
    var datos = { "userid": localStorage.getItem('user_id') };  
    console.log(datos);  
    $.ajax({  
        url: "http://138.100.72.88/hsc/incidents/getincidents.php",  
        type: "POST",  
        async: false,  
        dataType: 'json',  
        data: JSON.stringify(datos),  
        processData: false,  
        success: function (data) {  
            if (data.status == 'KO') {  
                alert("algo ha salido mal");  
            }  
            if (data.status == 'OK') {  
                console.log(data);  
                var elementosenviar = createIncidents(data);  
                console.log(elementosenviar);  
                return elementosenviar;  
            }  
            else {  
                console.log(data);  
            }  
        }  
    });  
}
```

Ilustración 22: Código AJAX de obtención de incidencias del usuario

```
function createIncidents(data) {  
    var elementoss = [];  
    data.data.forEach(function (element) {  
        var fecha1 = element.date_incident;  
        var fecha2 = fecha1.substring(0, fecha1.length - 9);  
        var incidenciaux = new incidencia(element.incident_title, fecha2, element.informed, element.closed)  
        elementoss.push(incidenciaux);  
    });  
    $scope.elementos = elementoss;  
  
    return elementoss;  
}
```

Ilustración 23: Código JavaScript de creación de incidencias

5.1.5 Encuestas

Las diferentes opciones de interacción se ven representadas en el siguiente

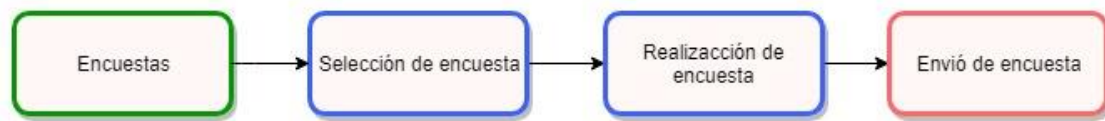


Ilustración 24: Diagrama de sección Encuestas

Al acceder el usuario a esta página, se le presentarán las posibles encuestas que puede realizar, para ellos se ha creado una lista mediante un `<ng-repeat>`, que recorre un objeto de clase `encuesta`, que contiene los parámetros correspondientes:

```
//Clase Encuesta
var encuesta = function (titulo,survey_model, json) {
  this.titulo = titulo;
  this.survey_model = survey_model;
  this.json = JSON.stringify(json);
}
```

Ilustración 25: Código de clase encuesta

Las diferentes encuestas disponibles para el usuario se obtienen de la base de datos mediante una llamada Ajax que podemos ver a continuación:

```
function getSurveys() {
  var datos = {};
  console.log(datos);
  $.ajax({
    url: "http://138.100.72.88/hsc/surveymodels/getsurveymodels.php",
    type: "POST",
    dataType: 'json',
    data: JSON.stringify(datos),
    processData: false,
    success: function (data) {
      console.log(data);
      if (data.status == 'KO') {
        alert("algo ha salido mal");
      }
      if (data.status == 'OK') {
        console.log(data);
        var elementosenviar = createSurveys(b);
        $scope.$apply(function () {
          $scope.elementos = elementosenviar;
          console.log('yoppp');
          console.log($scope.elementos);
        });
        return true;
      }
      else {
        console.log(data);
      }
    }
  });
}
```

Ilustración 26: Código AJAX de obtención de encuestas

Capítulo 5: Desarrollo de la aplicación

Una vez que se han obtenido las diferentes encuestas, estas se crean a partir del JSON obtenido a partir de la clase *encuesta* especificada anteriormente

```
function createSurveys(data) {
    console.log(data);
    var elementoss = [];
    data.data.forEach(function (element) {
        var encuestaux = new incidencia(element.title, element.survey_model_id, elemento.survey);
        elementoss.push(encuestaux);
    });
    $scope.elementos = elementoss;
    console.log($scope.elementos);

    return elementoss;
}
```

Ilustración 27: Código JavaScript de creación de encuestas

Dichas encuestas se pasan al HTML que tenemos a continuación gracias a que la variable *elementos* es compartida desde el controlador gracias a la directiva *\$scope*.

```
<div id="selectencuesta" >
    <ul class="list-group">
        <button ng-repeat="encuesta in elementos" ng-click="displaySurvey(encuesta.json)" type="button"
            class="list-group-item list-group-item-action active" style="margin-top:1%; background-color:dodgerblue">
            {{encuesta.titulo}}
        </button>
    </ul>
</div>
<div id="surveyContainer" style="display:none"></div>
```

Ilustración 28: Código HTML de selección de encuestas

En el momento que se selecciona una encuesta el *surveyContainer* aparece para poder realizar la encuesta correspondiente, cuenta con el JSON ya que ha sido recibido a través de la función llamada al clicar en la encuesta deseada, *displaySurvey()*.

```
$scope.displaySurvey = function (survey) {
    var surveyJSONsent = JSON.parse(survey);
    document.getElementById("surveyContainer").style.display = "block";
    document.getElementById("selectencuesta").style.display = "none";
    document.getElementById("titulo").style.display = "none";
    var survey = new Survey.Model(surveyJSONsent);
    $("#surveyContainer").Survey({
        model: survey,
        onComplete: sendDataToServer
    });
}
```

Ilustración 29: Código JavaScript de presentación de encuesta

Finalmente, al terminar la encuesta, como se observa en la imagen anterior, se llama a la función *sendDataToServer()* donde se realiza la llamada Ajax a través de la función *sendSurveyAjax*:

```
function sendSurveyAjax(survey) {
    var datos = { "userid": localStorage.getItem('user_id'), "survey_model": $scope.selectedSurveyModel, "answer": JSON.stringify(survey.data)};
    console.log(datos);
    $.ajax({
        url: "http://138.100.72.88/hsc/surveys/sendsurvey.php",
        type: "POST",
        dataType: 'json',
        data: JSON.stringify(datos),
        processData: false,
        success: function (data) {
            if (data.status == 'KO') {
                alert("algo ha salido mal");
            }
            if (data.status == 'OK') {
                console.log(data);
                return true;
            }
            else {
                console.log(data);
            }
        }
    });
}
```

Ilustración 30: Código AJAX de envío de encuestas

5.1.6 Servicios

En el siguiente diagrama se puede ver de manera visual la función y posibilidades de esta página:

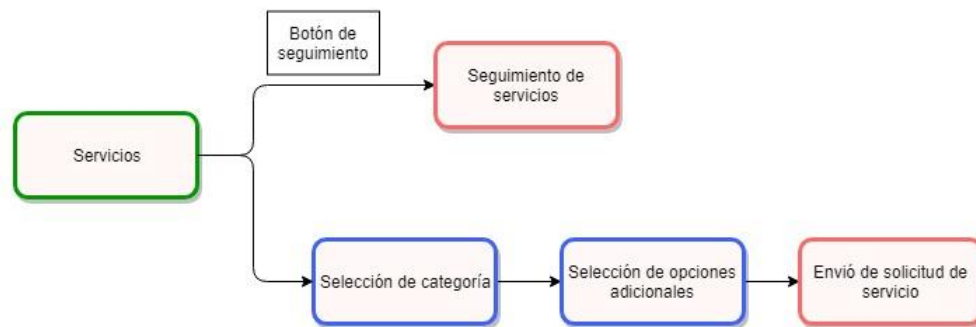


Ilustración 31: Diagrama de sección Servicios

En la sección de servicios, el usuario debe elegir el servicio que desea solicitar, y en el caso de que sea pertinente, deberá seleccionar otras opciones dentro del mismo. También tiene la posibilidad de añadir información adicional a través de un cuadro de texto.

Primero se selecciona un servicio, a través de un elemento *select*, una vez elegido, el bloque correspondiente de selección de opciones adicionales y envío del servicio aparece gracias a una función de JavaScript que se especifica a continuación:

```
<!-- Título bienvenida -->
<div style="text-align: center;">
  <div style="display: inline-block; width: 70%;">
    <!-- Título bienvenida -->
    <!-- Texto explicativo -->
    <div class="card border-primary" style="margin-top:20px; background-color:dodgerblue">
      <div class="card-body text-primary">
        <p class="card-text" style="font-size:large; margin:-15px; justify-content:flex-end; background-color:dodgerblue;color:white">Seleccione el servicio:</p>
      </div>
    </div>
    <!-- Selección de servicio -->
    <div class="form-group input-group">
      <div class="input-group-prepend">
        <span class="input-group-text"> <i class="fa fa-clipboard-list" style="margin-right:10px"></i> </span>
      </div>
      <select id="ServicioSeleccionado" class="form-control" required>
        <option value="" hidden></option>
        <option ng-model="pelu" ng-init="pelu=true">Peluqueria</option>
        <option>Biblioteca</option>
        <option>Quiloto</option>
        <option>Asistencia Religiosa</option>
        <option>Voluntariado</option>
        <option>Dieta</option>
      </select>
    </div>
  </div>
</div>
```

Ilustración 32: Código HTML de selección de servicio

```
<div id="bloquePeluqueria" style="display:none;">
  <canvas height="15%"></canvas>
  <div style="width:90%;margin: 0 auto">
    <textarea class="textareatro" id="textPeluqueria" rows="4" cols="35" placeholder="Especifique aquí los detalles"></textarea>
  </div>
</div>
<div id="bloqueBiblioteca" style="display:none;">
  <canvas height="15%"></canvas>
  <div class="form-group input-group">
    <div class="input-group-prepend">
      <span class="input-group-text"> <i class="fa fa-book" style="margin-right:10px"></i> </span>
    </div>
    <select id="Genero" class="form-control" required>
      <option value="" hidden>Género Literario</option>
      <option>Infantil</option>
      <option>Cómics y manga</option>
      <option>Intriga</option>
      <option>Historia</option>
      <option>Ya se verá</option>
    </select>
  </div>
  <div style="width:90%;margin: 0 auto">
    <textarea class="textareatro" id="textBiblioteca" rows="4" cols="35" placeholder="Especifique aquí los detalles"></textarea>
  </div>
</div>
```

Ilustración 33: Código HTML bloques de servicios

```
//Cuando cambia alguna "select" comprueba si puede activar el boton de enviar
$(document).ready(function () {
    $("select").change(function () {
        var ServicioSeleccionado = $("#ServicioSeleccionado").val();

        if (ServicioSeleccionado == "Peluqueria") {
            document.getElementById("bloquePeluqueria").style.display = "block";
            document.getElementById("bloqueBiblioteca").style.display = "none";
            document.getElementById("bloqueQuiosco").style.display = "none";
            document.getElementById("bloqueAsistencia").style.display = "none";
            document.getElementById("bloqueVoluntariado").style.display = "none";
            document.getElementById("bloqueDieta").style.display = "none";
            document.getElementById("bloqueEnviar").style.display = "block";
            document.getElementById("#boton-enviar").disabled = false;
            document.getElementById("textTermina").style.display = "none";
        }
    });
});
```

Ilustración 34: Código JavaScript de visualización de bloques de servicios

Las diferentes opciones en el caso del servicio de comidas han sido desarrolladas mediante un `<ng-repeat>` y un objeto de la clase correspondiente para almacenar los datos necesarios.

```
//Clase opcion
var opcion = function (index,nombreOpcion, info) {
    this.nombreOpcion = nombreOpcion;
    this.info = info;
    this.index = index;
}

//Clase dieta
var dieta = function (index,nombreDieta,info, ...opciones) {
    this.index = index;
    this.nombreDieta = nombreDieta;
    this.info = info;
    this.opciones = opciones;
}

//Creo las opciones y dieta
var m1 = new opcion(0,"Vegetariana", "Quinoa <br/> Tofu");
var m2 = new opcion(1,"Cetogenica", "Aguacate <br/> Chuletón");
var m3 = new opcion(2,"Paleo", "Conejo <br/> Patata");
var p1 = new opcion(0,"Dieta blanca", "Arroz <br/> Pollo");
var p2 = new opcion(1,"Puré", "Puré verduras <br/> Yogurt natural")

var dietaNone = new dieta(0, "", "", new opcion("", ""));
var dieta1 = new dieta(0,"Jovenes", "", m1, m2, m3);
var dieta2 = new dieta(1,"Persona mayor", "", p1, p2);

//En esta variable de almacenan todos los bloques y opciones
$scope.elementos = [dieta1, dieta2];
```

Ilustración 35: Código de clases y creación de opciones de dieta

Al igual que en la sección de incidencias al hacer clic en “Enviar” aparece un *pop-up* con la opción de cancelar el envío o confirmarlo, ha sido desarrollado de igual manera a través de una llamada Ajax, teniendo en cuenta que, dependiendo del servicio, se envían unos datos u otros.

```
function sendServiceAjax(user, servicio, genero, dieta, texto) {
  //data '{"userid":"9","type_of_service":"Biblioteca","genre_library":"Historia", "text_info": "Texto"
  if (servicio == 'Biblioteca') {
    var datos = { "userid": user, "type_of_service": servicio, "genre_library": genero, "text_info": texto };
  }
  else if (servicio == 'Dieta') {
    var datos = { "userid": user, "type_of_service": servicio, "option_diet": dieta, "text_info": texto };
  }
  else {
    var datos = { "userid": user, "type_of_service": servicio, "text_info": texto};
  }

  console.log(datos);
  $.ajax({
    url: "http://138.100.72.88/hsc/services/sendservice.php",
    type: "POST",
    dataType: 'json',
    data: JSON.stringify(datos),
    processData: false,
    success: function (data) {
      if (data.status == 'KO') {
        alert("Va se ha enviado este incidente");
      }
      if (data.status == 'OK') {
        console.log(data);
        return true;
      }
      else {
        console.log(data);
      }
    }
  });
}
```

Ilustración 36: Código AJAX de envío de servicio

Es esta página, también se puede acceder al seguimiento de servicios, aquí, al usuario se le presenta una tabla con los servicios que ha solicitado y el servicio de los mismos, ha sido desarrollado siguiendo el mismo enfoque por lo que no es necesario especificar el código.

5.1.7 Ajustes

Aquí se pueden realizar varias acciones, las cuales se pueden observar aquí:

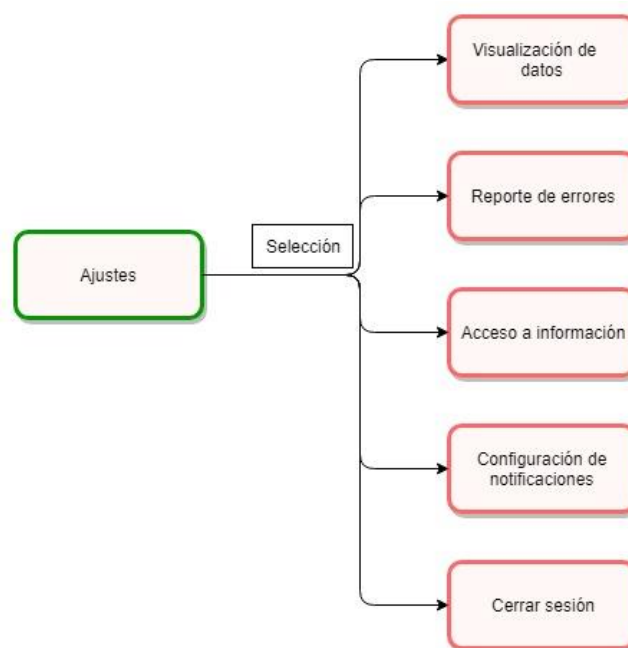


Ilustración 37: Diagrama de sección Ajustes

La estructura de esta página se compone de diferentes tarjetas y sub-tarjetas que se pueden expandir al clicar para revelar información u opciones adicionales dentro de las mismas, esto se realiza mediante la siguiente función:

```
//Esta función se encarga de hacer desaparecer y aparecer los DETALLES de los apartados
$scope.toggleajustes = function (index) {
    var nombreclase = "ajustes" + index;
    var x = document.getElementsByClassName(nombreclase);
    try {
        for (var i = 0, length = x.length; i < length; i++) {
            if (x[i].style.display === "none") {
                x[i].style.display = "block";
            } else {
                x[i].style.display = "none";
            }
        }
    } catch(err){}
}
```

Ilustración 38: Código JavaScript de visualización de bloques de ajustes

Con el botón de cerrar sesión se llama a una función que se encarga de borrar los datos almacenados de sesión y dirigir a la página de inicio de sesión.

```
//Se encarga de CERRAR SESIÓN
$scope.logout = function () {
    localStorage.setItem('signedIn', 'No');
    localStorage.setItem('Perfil', '');
    localStorage.setItem('Area', '');
    localStorage.setItem('Apellido', '');
    localStorage.setItem('Hab', '');
    localStorage.setItem('Aux', '');
    window.location.href = "index.html";
}
```

Ilustración 39: Código JavaScript de cierre de sesión

5.2 Base de datos

El proceso de desarrollo de la base de datos consta de tres etapas claramente diferenciadas, el diseño y creación de la misma, la subida de datos de prueba para poder comprobar el buen funcionamiento de la aplicación, por último, la lógica desarrollada en PHP, la cual han sido programadas por el tutor del trabajo.

5.2.1 Diseño de la BBDD

Como se ha comentado en el apartado de tecnologías utilizadas, el diseño se realizó en la plataforma web *dbdiagram.io*, por su fácil programación y exportación a MySQL, el cual es compatible con MariaDB, el gestor que utilizado para base de datos.

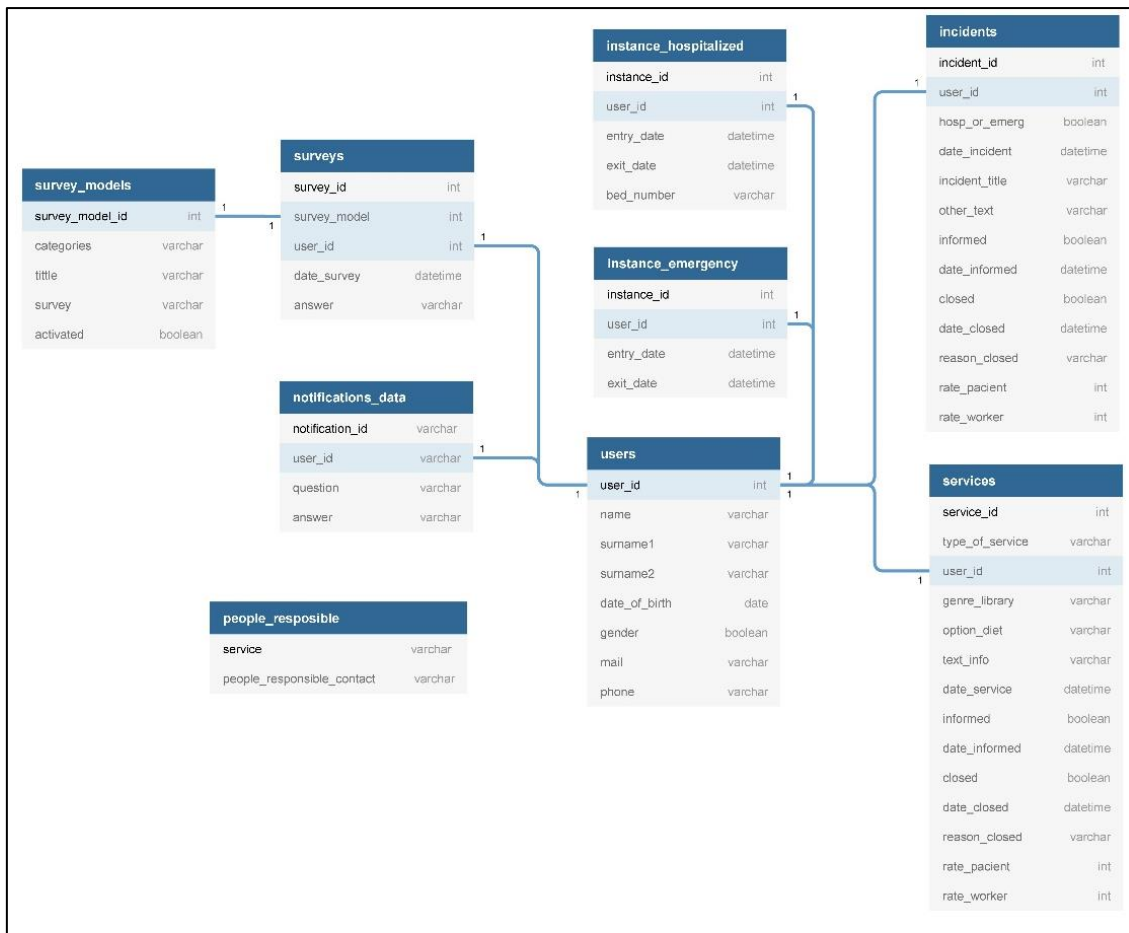


Ilustración 40: Diagrama de tablas de la base de datos

Todas las relaciones de la base de datos son relaciones *one-to-many*, hace referencia a la relación entre dos entidades, A y B, en las cuales un elemento de A puede estar conectado a varios elementos de B, pero un miembro de B solo puede estar conectado a un elemento de A. Por ejemplo, si pensamos en A como libros, y B como páginas. Un libro puede tener muchas páginas, pero una página solo puede tener un libro.

Aquí tenemos las diferentes tablas, pasaremos a explicar cada una a continuación:

Tabla de pacientes: denominada *users*, cuenta con los siguientes campos: *user_id* (identificador del paciente), *name* (nombre), *surname1* (primer apellido), *surname2* (segundo apellido), *date_of_birth* (fecha de nacimiento), *gender* (género), *mail* (email de contacto) y *phone* (teléfono de contacto).

user_id	name	surname1	surname2	date_of_birth	gender	mail	phone
1	Miguel	Cordero	Collar	1995-09-29	0	miguelcorderocollar@gmail.com	+34606912528
2	Jane	Doe	Lopez	2000-09-29	0	janedoelopez@gmail.com	+34606912528

Ilustración 41: Tabla users

Tabla de incidencias: denominada *incidents*, cuenta con los siguientes campos: *incident_id* (identificador del incidente), *user_id* (identificador del usuario al que pertenece la incidencia), *hosp_or_emerg* (valor binario que indica si es una incidencia de un paciente hospitalizado o de urgencias), *date_incident* (fecha y hora del incidente), *incident_title* (título identificativo de la incidencia), *other_text* (texto explicativo del incidente), *informed* (valor binario que indica si ha sido informada la persona responsable), *date_informed* (fecha y hora a la que se ha informado), *closed* (valor binario que indica si se ha cerrado la incidencia), *date_closed* (fecha y hora a la que se ha cerrado), *reason_closed* (razón de cierre de incidencia), *rate_patient* (calificación de satisfacción del paciente) y *rate_worker* (calificación de satisfacción del trabajador).

incident_id	user_id	hosp_or_emerg	date_incident	incident_title	other_text	informed	date_informed	closed	date_closed	reason_closed	rate_patient	rate_worker
1	12	1	2019-04-25 21:57:19	Mobiliario Negativo	Otro texto	1	2019-04-25 21:57:19	0	NULL	NULL	NULL	NULL
2	7	1	2019-01-01 19:02:01	Asistencia Deficiente	Otro texto	1	2019-01-01 19:02:01	0	NULL	NULL	NULL	NULL
3	18	1	2019-03-11 19:13:44	Limpieza Positiva	Otro texto	1	2019-03-11 19:13:44	1	2019-03-11 23:37:37	Solucionado	8	9
4	7	0	2019-03-16 02:06:02	Comida Positiva	Otro texto	1	2019-03-16 02:06:02	1	2019-03-16 22:37:10	Todo correcto	9	8
5	16	0	2019-04-04 04:32:08	Limpieza Deficiente	Otro texto	1	2019-04-04 04:32:08	1	2019-04-04 23:05:21	Arreglado	8	6

Ilustración 42: Tabla incidents

Tabla de entradas de hospitalización: denominada *instances_hospitalization*, cuenta con los siguientes campos: *instance_id* (identificador de la entrada), *user_id* (usuario), *entry_date* (fecha y hora de entrada), *exit_date* (fecha y hora de salida) y *bed_number* (habitación del paciente)

instance_id	user_id	entry_date	exit_date	bed_number
1	20	2019-07-04 10:08:56	NULL	421
2	6	2019-07-02 11:16:01	2019-07-02 22:16:03	30

Ilustración 43: Tabla instances_hospitalization

Tabla de entradas de urgencias: denominada *instances_emergency*, cuenta con los siguientes campos: *instance_id* (identificador de la entrada), *user_id* (usuario), *entry_date* (fecha y hora de entrada) y *exit_date* (fecha y hora de salida)

instance_id	user_id	entry_date	exit_date
1	2	2019-06-30 12:48:59	NULL
2	10	2019-06-28 06:24:28	NULL

Ilustración 44: Tabla instances_emergency

Tabla de notificaciones: denominada *notification_data*, cuenta con los siguientes campos: *notification_id* (identificador de la notificación), *user_id* (usuario), *question* (pregunta recibida) y *answer* (respuesta).

<i>notification_id</i>	<i>user_id</i>	<i>question</i>	<i>answer</i>
1	13	¿Le gustaría hacer una encuesta?	No
2	19	¿Tiene alguna sugerencia?	En otro momento

Ilustración 45: Tabla *notification_data*

Tabla de servicios: denominada *services*, cuenta con los siguientes campos: *service_id* (identificador del servicio), *type_of_service* (el tipo de servicio correspondiente), *user_id* (identificador del usuario al que pertenece el servicio), *genre_library* (en el caso de ser servicio de biblioteca, el género literario), *option_diet* (en el caso de ser servicio de dieta, la opción elegida), *text_info* (texto explicativo del servicio), *informed* (valor binario que indica si ha sido informada la persona responsable), *date_informed* (fecha y hora a la que se ha informado), *closed* (valor binario que indica si se ha cerrado la incidencia), *date_closed* (fecha y hora a la que se ha cerrado), *reason_closed* (razón de cierre de incidencia), *rate_patient* (calificación de satisfacción del paciente) y *rate_worker* (calificación de satisfacción del trabajador).

<i>service_id</i>	<i>type_of_service</i>	<i>user_id</i>	<i>genre_library</i> <small>(en el caso de ser servicio de biblioteca)</small>	<i>option_diet</i> <small>(en el caso de ser servicio de dieta)</small>	<i>text_info</i>	<i>date_service</i>	<i>informed</i>	<i>date_informed</i>	<i>closed</i>	<i>date_closed</i>	<i>reason_closed</i>	<i>rate_patient</i>	<i>rate_worker</i>
1	Peluquería	5	NULL	NULL	Otro text	2019-05-24 02:44:34	1	2019-05-24 02:44:34	1	2019-05-24 23:35:46	Solucionado	5	9
2	Dieta	11	NULL	Dieta Proteica	Otro text	2019-06-09 01:40:26	1	2019-06-09 01:40:26	0	NULL	NULL	NULL	NULL
3	Asistencia Religiosa	8	NULL	NULL	Otro text	2019-06-08 17:50:38	1	2019-06-08 17:50:38	1	2019-06-08 23:03:53	Arreglado	7	7
4	Biblioteca	4	Intriga	NULL	Otro text	2019-05-15 09:12:25	1	2019-05-15 09:12:25	1	2019-05-15 22:09:59	Arreglado	7	6
5	Voluntariado	11	NULL	NULL	Otro text	2019-06-05 16:38:40	1	2019-06-05 16:38:40	0	NULL	NULL	NULL	NULL

Ilustración 46: Tabla *services*

Tabla de modelos de encuesta: denominada *survey_models*, cuenta con los siguientes campos: *survey_model_id* (identificador del modelo de encuesta), *categories* (categorías a las que va destinada la encuesta), *title* (título de la encuesta), *survey* (encuesta en formato JSON) y *activated* (valor binario que indica si esta activa dicha encuesta).

<i>survey_model_id</i>	<i>categories</i>	<i>title</i>	<i>survey</i>	<i>activated</i>
1	NULL	Encuesta Básica	{ pages: [{ name: "page1", elements: [{ type: "rat...	1
2	NULL	Encuesta Básica v2	{ pages: [{ name: "page1", elements: [{ type: "ra...	1

Ilustración 47: Tabla *survey_models*

Tabla de encuestas: denominada *surveys*, cuenta con los siguientes campos: *survey_id* (identificador de la encuesta), *survey_model* (modelo de encuesta), *user_id* (usuario que ha realizado la encuesta), *date_survey* (fecha y hora) y *answer* (respuesta).

<i>survey_id</i>	<i>survey_model</i>	<i>user_id</i>	<i>date_survey</i>	<i>answer</i>
1	1	1	2019-04-19 22:23:33	{"Pregunta N° 1":5,"question1":5,"question2":4,"qu...
2	1	1	2019-03-19 22:23:33	{"Pregunta N° 1":5,"question1":5,"question2":4,"qu...
3	1	1	2019-05-19 22:23:33	{"Pregunta N° 1":5,"question1":5,"question2":4,"qu...

Ilustración 48: Tabla *surveys*

5.2.2 Población de la BBDD

La existencia de datos de prueba es fundamental, para crear un suficiente número de datos, y que estos sean tanto aleatorios como con sentido, la estrategia elegida fue la de programar la creación del código SQL para aportar flexibilidad.

El lenguaje elegido, como se ha comentado anteriormente sería C#, y los *scripts* fueron realizados en la plataforma online *.NET Fiddle* para su rápido desarrollo, ya que era algo simple lo que se requería.

Para esto generé un script para prácticamente cada tabla, en el cual se partía de un *string* base, que incluía la parte repetida de las *queries*, donde se indican los valores que a continuación se van a aportar y el orden de los mismos. La segunda parte era la generada automáticamente a través de una aleatorización de una serie de valores y de manera repetida a través de un bucle *for*.

Aquí un ejemplo del script más básico realizado, en la tabla *notification_data*:

```
1 using System;
2
3 public class Program
4 {
5     public static void Main()
6     {
7         var base1 = "INSERT INTO notifications_data (user_id, question, answer) VALUES (";
8         string[] preguntas = {"¿Le gustaría hacer una encuesta?", "¿Tiene alguna sugerencia?"};
9         string[] respuestas = {"Si", "No", "En otro momento"};
10        var stringfinal = "";
11        Random gen = new Random();
12        for (int i = 0; i < 20; i++)
13        {
14            stringfinal = base1;
15            var otro = "1, '¿Le gustaría hacer una encuesta?', 'Si'";
16            stringfinal = stringfinal + gen.Next(1, 21) + ", '" + preguntas[gen.Next(2)] + "', '" + respuestas[gen.Next(3)] + "'";
17            Console.WriteLine(stringfinal);
18        }
19    }
20 }
```

Ilustración 49: Código script creación de SQL

5.2.3 Comunicación mediante Ajax

La manera de comunicarse con las funciones lógicas creadas, y por tanto la base de datos, es mediante el método Ajax incluido en la librería de jQuery, que permite que un servidor y una aplicación puedan intercambiar información de manera asíncrona (llamadas que pueden ser cumplidas en el momento o futuro).

Mediante este método, se realiza envía y recibe la información, y se interpreta la respuesta. Otra funcionalidad interesante es el hecho de que se puede modificar la vista HTML para informar por ejemplo de algo sin necesidad de recargar la página.

Para el método Ajax elegido hay varios parámetros que se deben definir, no obstante, hay muchos más que no han sido utilizados:

- La *url* a la que se envía la petición, siendo esta la que almacena la lógica programada en PHP.
- El tipo de petición(*type*), ya que esta puede ser GET (recepción de información) o POST (envío de información).
- El *data*, que es la información que se desea enviar al servidor, en nuestro caso es un JSON.

- El *dataType*, donde se especifica el tipo de datos que se espera recibir por parte del servidor, es nuestro caso JSON al igual que el envío.
- En el parámetro *success* se especifica cómo se debe proceder si se recibe una respuesta satisfactoria por parte del servidor, todo esto en JavaScript.
- El último parámetro utilizado es *error*, el cual realiza una función similar a la del parámetro anterior, pero en este caso, es para códigos de errores proporcionados por el servidor.

5.2.4 Lógica PHP de la BBDD

Hacen falta varias funciones que debe realizar la base de datos para hacer funcional la aplicación, estas han sido programadas en PHP por el tutor del TFG, Joaquín Ordieres, su dirección es dual, tanto enviar información a la aplicación como recibir información para almacenarla en la base de datos.

CheckLogin

Función: Se encarga de validar que el usuario se encuentra en la base de datos y que hay una entrada abierta, ya sea para hospitalizados o urgencias, según lo que haya dicho y se comparan los valores de verificación.

Se envía: *area* (hospitalizado o urgencias), *apellido*, *fechanac* (fecha de nacimiento) y *hab* (Nº de habitación) en el caso de que el área sea hospitalizado o *genero* (Género) en caso contrario.

Se recibe: confirmación junto con el identificador del usuario o el error identificando lo que no coincide.

SendIncident

Función: Se encarga de guardar en la base de datos un incidente y envía correo al responsable correspondiente.

Se envía: *userid* (identificador del usuario), *hospreme* (valor binario que indica el área a la que pertenece la queja), *title* (título de la incidencia), *other* (texto correspondiente a otras opciones no especificadas).

Se recibe: confirmación de que ha salido todo correcto o del error correspondiente.

GetIncidents

Función: Se encarga de enviar a la aplicación el historial de incidencias del usuario que lo solicita.

Se envía: *user_id* (identificador del usuario)

Se recibe: si todo sale correcto, las incidencias con sus datos en formato JSON, y si no, el error correspondiente.

GetSurveys

Función: Se encarga de enviar a la aplicación las encuestas disponibles para el usuario.

Se envía: *user_id* (identificador del usuario)

Se recibe: si todo sale correcto, las encuestas, con sus datos en formato JSON, y si no, el error correspondiente.

SendSurveys

Función: Se encarga de almacenar en la BBDD la encuesta enviada por la aplicación.

Se envía: *userid* (identificador del usuario), *survey_model* (identificador de la encuesta realizada) y *answer* (respuesta a la encuesta en formato JSON).

Se recibe: confirmación de que todo ha salido correcto o el error identificando el problema

SendService

Función: Se encarga de guardar en la base de datos un servicio e informar por mail a la persona correspondiente.

Se envía: *userid* (identificador del usuario), *type_of_service* (servicio solicitado), *text_info* (información complementaria del servicio), *genre_library* (género de libro) en el caso de que se trate de un servicio de biblioteca y *option_diet* (dieta elegida) en el caso de que se trate de un servicio de dieta.

Se recibe: confirmación de que todo ha salido correcto y en caso contrario, el código de error identificando el problema.

GetServices

Función: Se encarga de enviar a la aplicación el historial de servicios solicitados del usuario.

Se envía: *user_id* (identificador del usuario)

Se recibe: si todo sale correcto, los servicios con sus datos en formato JSON, y si no, el error correspondiente.

SendNotification

Función: Se encarga almacenar en la base de datos una notificación.

Se envía: *userid* (identificador del usuario), *questions* (pregunta realizada) y *answer* (respuesta dada por el usuario).

Se recibe: confirmación de que ha salido correcto o código de error correspondiente con el problema.

5.3 Notificaciones

La principal utilidad que se le va a dar a las notificaciones en este prototipo, consiste en el recordatorio de realización de encuestas, para animar a aumentar la cantidad de *feedback* recibido y poder aportar un mejor servicio teniendo en cuenta la información obtenida.

Para realizar las notificaciones, se ha utilizado el plugin *cordova-plugin-local-notifications*, el cual tiene una licencia Apache 2.0, permitiendo su uso, ya que es de código abierto.

Se ha configurado una notificación preguntando al usuario si le importaría realizar una encuesta, dándole tres opciones: “Sí”, “No” y “En otro momento”, en el caso de la persona clique “Sí”, accederá a la pantalla correspondiente para realizar la encuesta.

La notificación aparecerá una vez a la semana hasta tres ocasiones, y en el caso de que haya realizado la encuesta, dicha notificación se cancela para no molestar al usuario.

```
document.addEventListener('deviceready', function () {

    cordova.plugins.notification.local.hasPermission(function (granted) { });

    cordova.plugins.notification.local.schedule({
        id: 1,
        title: '¿Le importaría realizar una encuesta?',
        actions: [
            { id: 'yes', title: 'Si' },
            { id: 'no', title: 'No' },
            { id: 'other', title: 'En otro momento' },
        ],
        firstAt: next_monday,
        trigger: { every: 'week', count: 3 }
    });

    cordova.plugins.notification.local.on('yes', function (notification) {
        if (notification.id === 1) {
            window.location.href = "index.html#!/selectencuesta";
            localStorage.setItem('notificationYes', 'Yes');
        }
    });

    if (localStorage.getItem('signedIn') === 'Yes') {
        cordova.plugins.notification.local.cancel(1, function () {}, scope);
    }

});
```

Ilustración 50: Código de creación notificaciones

Obteniéndose el siguiente resultado:

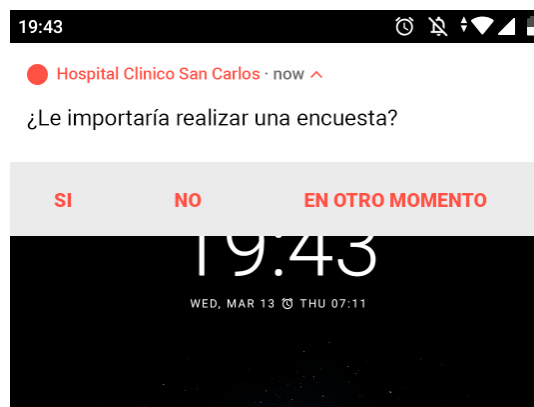


Ilustración 51: Captura de ejemplo de notificación

El usuario al responder a la notificación, se realiza el envío de la pregunta y envío de la respuesta por medio de una llamada Ajax:

```
function sendNotification(notification) {  
    var datos = { "userid": localStorage.getItem('user_id'), "questions": notification.title, "answer": notification.answer };  
    console.log(datos);  
    $.ajax({  
        url: "http://138.100.72.88/hsc/notifications/sendnotification.php",  
        type: "POST",  
        dataType: 'json',  
        data: JSON.stringify(datos),  
        success: function (data) {  
            if (data.status == 'KO') {  
                alert("algo ha salido mal");  
            }  
            if (data.status == 'OK') {  
                console.log(data);  
                return true;  
            }  
            else {  
                console.log(data);  
            }  
        }  
    });  
}
```

Ilustración 52: Código AJAX de envío de notificaciones

5.4 Capturas de pantalla

Este apartado de la memoria vamos a ver el resultado final de experiencia del usuario a nivel visual.

Comenzamos con la página de inicio de sesión, donde el usuario debe rellenar una serie de campos para poder asegurar sus credenciales, el menú principal, donde el usuario puede seleccionar la página deseada y, por último, la barra de navegación abierta.

The image displays three sequential screenshots of a web application for 'Hospital San Carlos'. The first screenshot shows the login page with a 'Bienvenido' header, a descriptive text box, and input fields for 'Paciente', 'Hospitalizado', 'Apellido del paciente', and '29/09/1995', along with a checkbox for 'Acepto la Política de Privacidad' and an 'INICIAR SESIÓN' button. The second screenshot shows the main menu with the same header and text box, but with four large icons for 'Incidentes', 'Encuesta', 'Servicios', and 'Ajustes'. The third screenshot shows the main menu with a red sidebar on the right containing a list of options: 'MAIN', 'INCIDENCIAS', 'ENCUESTAS', 'SERVICIOS', and 'AJUSTES'. Below this list is a section for 'Area' with a text box, input fields for 'Apellido del paciente' and 'dd/mm/aaaa', a checkbox for 'Acepto la Política de Privacidad', and an 'INICIAR SESIÓN' button.

Ilustración 53: Capturas de sección de Inicio de sesión

Aplicación móvil para recabar la opinión de los pacientes sobre la calidad del servicio sanitario percibido

Pasamos al servicio de reporte de incidencias, donde podemos observar la página inicial, con los diferentes bloques y opciones dentro de los mismos, el botón de confirmación de envío y la página de seguimiento de incidencias reportadas anteriormente por el usuario.

The first screenshot shows the 'Incidencias' (Incidents) screen. It has a red header with 'Hospital San Carlos' and a menu icon. Below the header, there's a title 'Incidencias' and a green 'SEGUIMIENTO' button. The main area contains several blue buttons: 'Atención', 'Instalaciones', 'Temperatura', '? Subir la temperatura', 'Limpieza', 'Mobiliario', 'Comida', and 'Otra Incidencia'. At the bottom is a green 'ENVIAR' button.

The second screenshot shows a confirmation dialog: '¿Seguro que desea enviar esta incidencia?' with 'NO' and 'SI' options. Below it, the same incident reporting options are visible.

The third screenshot shows the 'Seguimiento de incidencias' (Incident Tracking) screen. It has a red header with 'Hospital San Carlos' and a menu icon. Below the header, there's a title 'Seguimiento de incidencias'. The main area is a table with columns: 'Fecha', 'Incidencia', 'Informado', and 'Cerrado'.

Fecha	Incidencia	Informado	Cerrado
05/05/19	Temperatura Alta	Sí	No
02/05/19	Limpieza	Sí	Sí
20/04/19	Error Comida	Sí	Sí
02/04/19	Mobiliario	Sí	Sí
15/03/19	Horario	Sí	Sí
13/03/19	Comida Alegria	Sí	Sí

Ilustración 54: Capturas de sección de Incidencias

Pasamos a la sección de encuestas, donde el usuario puede elegir la encuesta a realizar dentro de las correspondientes a su perfil y posteriormente rellenar la misma.

The first screenshot shows the 'Encuestas' (Surveys) screen. It has a red header with 'Hospital San Carlos' and a menu icon. Below the header, there's a title 'Encuestas'. The main area contains several blue buttons: 'Encuesta General', 'Encuesta sobre las Instalaciones', 'Encuesta sobre su Experiencia', and 'Encuesta de Servicio'.

The second screenshot shows the survey completion screen. It has a red header with 'Hospital San Carlos' and a menu icon. Below the header, there's a title 'Encuestas'. The main area contains four survey questions, each with a 5-point scale from 'MAL' to 'GENIAL'.

- ¿Qué le parecen nuestras instalaciones? *
- ¿Cuál es su opinión respecto a nuestro servicio? *
- ¿Qué opina usted de la cafetería? *
- ¿Cómo definiría la velocidad de atención? *

Ilustración 55: Capturas de sección de Encuestas

A continuación, tenemos capturas de pantalla de solicitud de servicios, y seguimiento de los servicios demandados por el usuario.

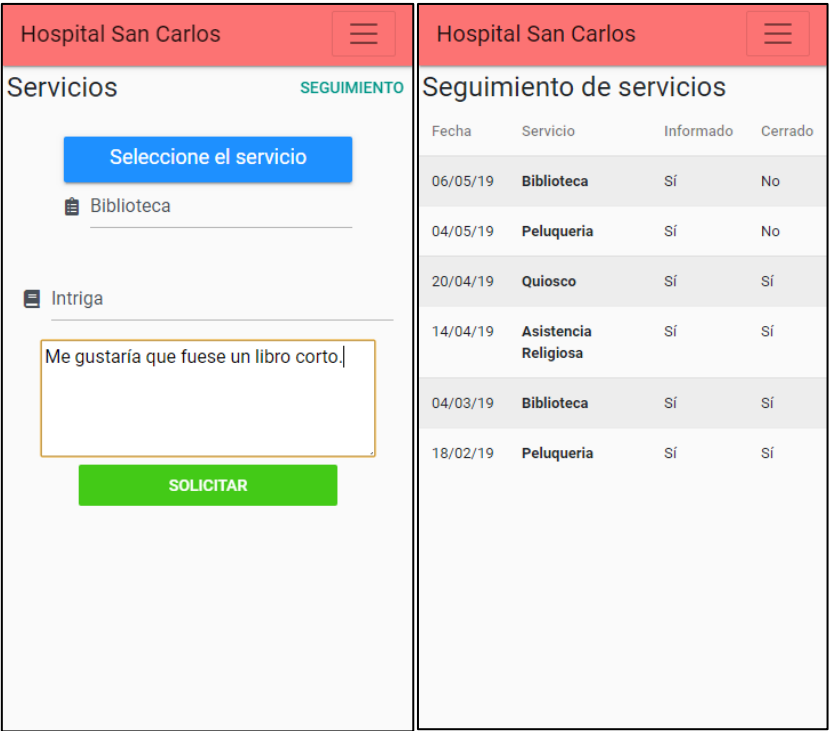


Ilustración 56: Capturas de sección de Servicios

Por último, el menú de ajustes, donde puede consultar los datos almacenados por la aplicación, acceder a información, configurar diferentes parámetros y cerrar la sesión.

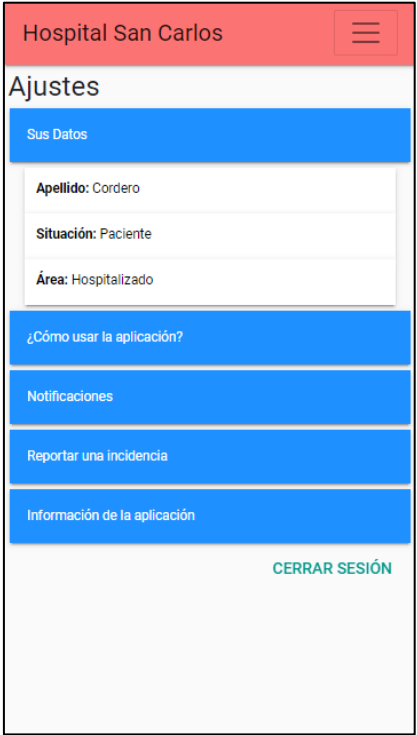


Ilustración 57: Captura de sección de Ajustes

CAPÍTULO 6: CONCLUSIONES Y LÍNEAS FUTURAS DE DESARROLLO

6.1 Valoración de impacto y aspectos de responsabilidad

Es importante tratar el tema de la normativa legal de aplicación, ya que hoy en día las cuestiones sobre la privacidad y la protección de datos son muy relevantes y puede acarrear importantes consecuencias legales si no hacen las cosas de acuerdo a la norma.

El uso de datos, en este caso de pacientes, está muy restringido en el ámbito de la sanidad, y en todo caso debe cumplir con la “Ley Orgánica de Protección de Datos Personales y garantía de los derechos digitales”, aprobada en diciembre del 2018 para cumplir con el “Reglamento General de Protección de Datos” (RGPD) aprobado por la Unión Europea. En esta normativa se estipula las diferentes tipologías de datos, y su posible utilización en los diferentes contextos, como se deben almacenar los datos, que datos se pueden solicitar, el fin siempre debe estar justificado para solicitar un dato, la persona debe estar plenamente informada y manifestar la conformidad, reparto de responsabilidades y reglas a seguir para el buen uso de los datos.

Para el prototipo de la aplicación, se han usado datos de prueba, datos no reales por lo tanto no ha sido necesario incluir una página de privacidad, pero en el caso de que se implementase la aplicación, el usuario tendría que tener la posibilidad de informarse del uso que van a tener sus datos personales (nunca datos médicos, ni relacionados con patologías), cuales datos y como se van a utilizar. Y para poder hacer uso de esos datos, el usuario debería darnos su conformidad y consentimiento.

El impacto que tendría la aplicación es totalmente positivo, ya que cumple una función muy necesaria, aliviando carga de trabajo a los servicios de enfermería, permitiendo que las incidencias se resuelvan con mayor celeridad, aportando información muy valiosa a la dirección del hospital sobre los principales aspectos a mejorar y facilita el acceso a múltiples servicios que pueden mejorar la experiencia en el hospital para los pacientes. Además, se ahorraría recursos materiales como son el papel, y el tiempo del servicio de enfermería.

6.2 Conclusiones

Teniendo en cuenta los resultados obtenidos y la información recabada a lo largo de la realización del trabajo, se ha llegado a las siguientes conclusiones:

- Existe un margen de mejora importante en la eficiencia de algunos aspectos del hospital, y la digitalización es una pieza clave para poder proveer de un mejor servicio, quitar carga de trabajo a determinados trabajadores para que puedan realizar su trabajo mejor.
- El marco de desarrollo Cordova es una gran herramienta para el desarrollo de aplicaciones de manera rápida y sencilla, con simples conocimientos de desarrollo web, siendo la mejor opción para aplicaciones sin demasiada complejidad, aportando mucho valor con poca inversión. Además, el avance en potencia computacional de los dispositivos móviles garantiza un gran rendimiento a pesar de la diferencia de optimización de este enfoque de desarrollo.
- Se han cumplido con los objetivos planteados inicialmente y se han implementado ideas que han ido surgiendo durante el desarrollo, además se han marcado unas líneas futuro de desarrollo claras y con mucho aporte de valor.
- El diseño de la base de datos es parte fundamental del desarrollo, ya que no solo debe proveer y almacenar información, debe facilitar la extracción de datos para poder hacer un buen análisis de la situación.
- Los beneficios que aporta la aplicación son amplios. Desde la más rápida atención de los pacientes, pasando por atender con mayor precisión las necesidades, hasta la solución de problemáticas con las que paciente se encuentra. El ahorro de costes del hospital es evidente, gracias al canal directo de comunicación se liberan recursos temporales en muchos trabajadores y permite aumentar la productividad. Por último, en análisis de datos que permite la base de datos es muy valioso para conocer el estado y necesidades del hospital.

6.3 Líneas futuras de desarrollo

Se han conseguido los objetivos propuestos de desarrollo, se ha realizado una prueba de concepto con un resultado positivo, peor durante el desarrollo, la idea ha ido evolucionando surgiendo nuevas ideas y funcionalidades que no se han podido incluir finalmente. A continuación, se especifican los posibles pasos a seguir:

- Programar una aplicación aparte, o dentro de la misma, en la que los responsables de los servicios/incidencias puedan recibir las tareas a realizar, cerrarlas y poder interactuar de manera más sencilla con el servidor.
- Hacer más flexible la aplicación, en el sentido de que se puedan añadir servicios o incidencias desde la base de datos en vez de programando.
- Desarrollar un panel de control para el hospital donde puedan introducir y modificar valores de la aplicación a la vez que ver poder analizar los datos recogidos de los diferentes apartados.
- Hacer pruebas para el sistema operativo iOS, ya que no ha sido posible realizarlas, adaptando los diferentes plugins en el caso de que sea necesario.
- Desarrollar la conexión de nuestra base de datos con la del hospital para hacerlo funcional, ya que sería necesario obtener los datos de los pacientes que se encuentran en el hospital, tanto hospitalizados y urgencias en cada momento.
- Mejoras de diseño de la aplicación, ya que, es algo que cuesta poco trabajo pero que es muy necesario para animar a los usuarios a utilizar la aplicación.
- Desarrollar un tutorial de introducción para la primera vez que se usa la aplicación para informar al usuario de las capacidades y funcionamiento de la misma.
- Implantar sistema de reporte de *bugs* y de problemas relacionados con el uso de la aplicación.

CAPÍTULO 7: PLANIFICACIÓN TEMPORAL Y PRESUPUESTO

7.1 Estructura de descomposición del proyecto

A la hora de dividir el proyecto en varias secciones claramente diferenciadas se ha decidido trabajar con la siguiente estructura. Estudios previos, diseño de la aplicación, desarrollo de la aplicación y redacción de la memoria.

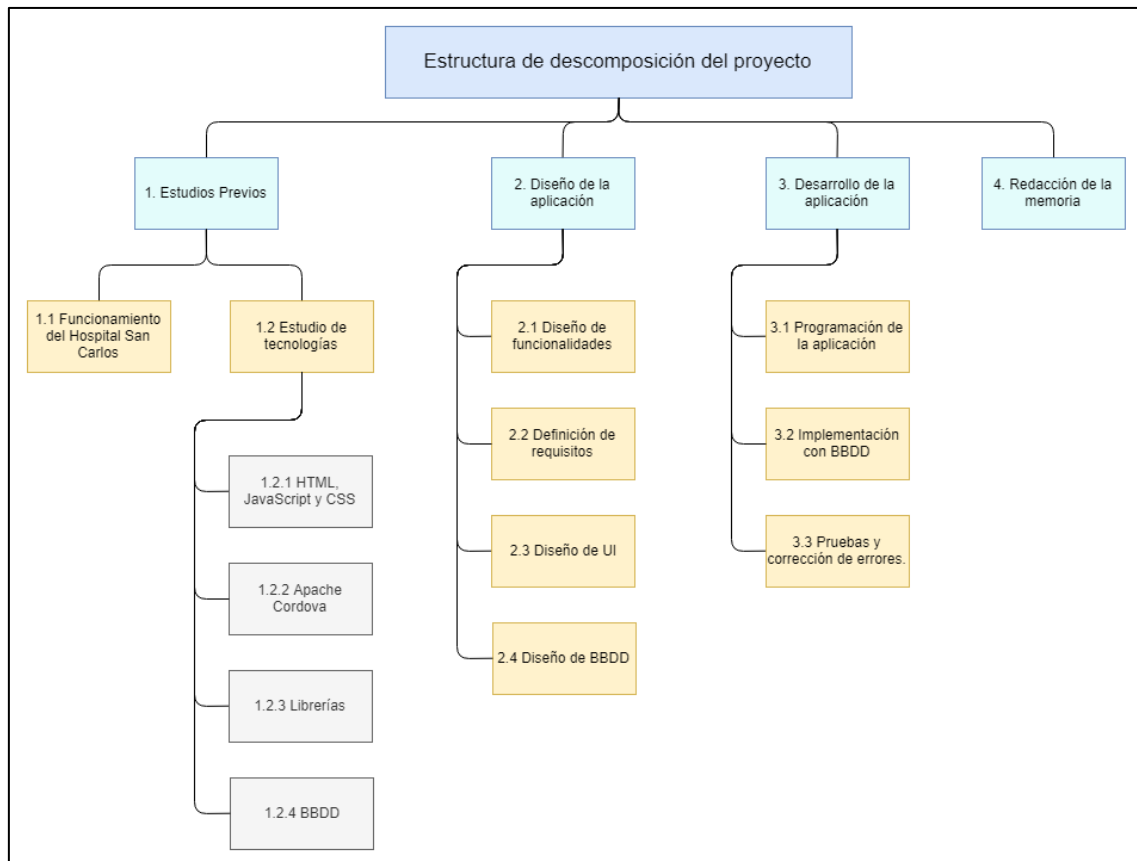


Ilustración 58: Estructura de descomposición del proyecto

7.2 Diagrama de Gantt

En el diagrama se puede observar como se ha repartido el tiempo entre las diferentes tareas del proyecto. Hay que tener en cuenta que la dedicación de horas no ha sido igual durante todas las etapas del proceso, principalmente por razones de disponibilidad.

La fecha de comienzo de trabajo se considera que fue el día 1 de diciembre de 2018, aunque la asignación de proyecto fue anterior, por motivos de disponibilidad no se pudo comenzar la parte de estudios previos hasta aproximadamente dicha fecha. El día 27 de febrero tuvo lugar la primera reunión con los tutores para empezar a diseñar la aplicación, requisitos y funcionalidades. La fecha de finalización fue el día 20 de junio, con una duración aproximada de seis meses.

Como se puede observar, la etapa que ha requerido un mayor número de horas ha sido la de desarrollo de la aplicación, ya que, en este tipo de proyectos, es la parte principal. La fase de estudios previos ha sido la siguiente en requerimiento de horas, principalmente por la inicial falta de conocimientos en la materia.

Esta distribución es orientativa y no representa fielmente el reparto de trabajo diario, ya que hay tareas que se han tenido que reanudar una vez finalizadas para realizar correcciones o cambios puntuales.

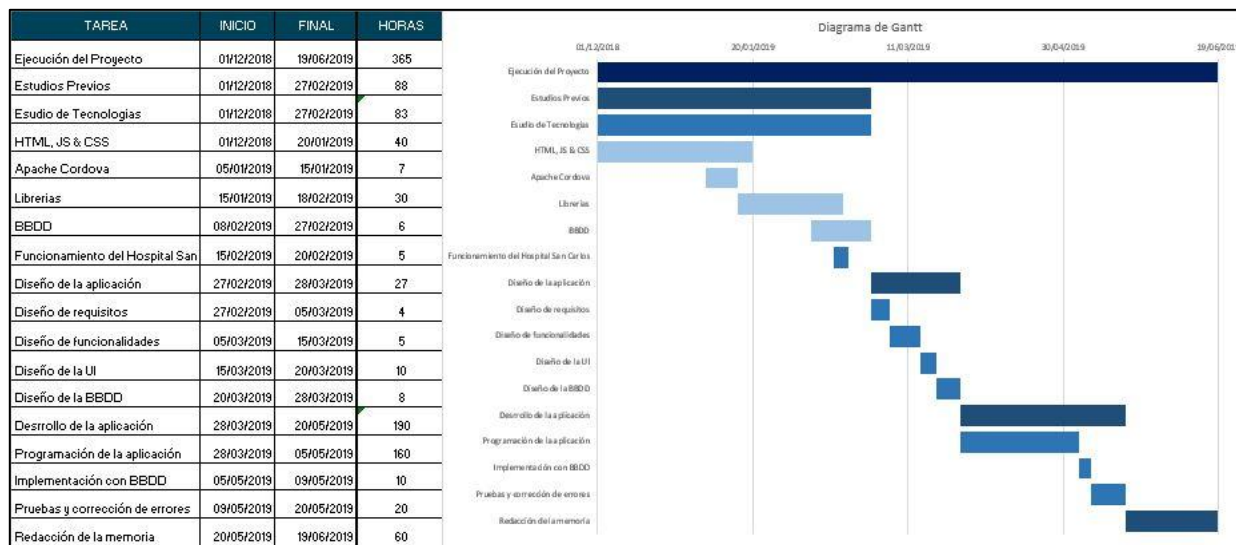


Ilustración 59: Diagrama de Gantt

7.3 Presupuesto

Para calcular el presupuesto del proyecto, se ha de tener en cuenta tres tipos de recursos, de hardware, humanos y software, los desgloses se encuentran a continuación. Dicho presupuesto hace referencia al coste si este proyecto hubiese sido desarrollado por un profesional a precios de mercado.

- Recursos de hardware: aquí se tiene en cuenta el gasto material de equipos y herramientas utilizados, a la vez que la amortización de los mismos, obtenida de la tabla de coeficientes de amortización lineal de la Agencia Tributaria.

Concepto	Coste (€)	Amortización	Total (€)
<i>Ordenador portátil</i>	700	10%	70
<i>Móvil Xiaomi</i>	200	10%	20
<i>Tableta Samsung</i>	180	10%	18
Total			108

Tabla 5: Costes de hardware

La amortización ha sido calculada como la correspondiente al periodo de tiempo del proyecto, teniendo en cuenta que la amortización anual es de 20%, por lo tanto, sería del 10%.

- Recursos humanos: Para los estudios previos y redacción de la memoria, estas tareas serian realizadas por un documentalista, el desarrollo por un programador y haría falta una supervisión del proyecto por un Ingeniero de Proyectos, por ejemplo.

Concepto	Tiempo dedicado (h)	Coste por hora (€/h)	Total (€)
<i>Ingeniero de Proyectos</i>	30	50	1500
<i>Documentalista</i>	60	15	900
<i>Programador</i>	115	20	2300
Total			4700

Tabla 6: Costes de recursos humanos

Hay que tener en cuenta que, al ser realizado por un profesional, la parte de estudio de tecnologías se ve reducida casi en su totalidad y la parte de desarrollo de acorta ya que una persona experta requiere de un menor tiempo para realizar la misma labor.

- Recursos de software: Al utilizar todo herramientas de software libre, el coste sería nulo, pero hay que tener en cuenta que, por ejemplo, para desplegar la aplicación en *App Store* de Apple por ejemplo tendría un coste.

Presupuesto Total

Concepto	Importe (€)
<i>Recursos de hardware</i>	108.00
<i>Recursos humanos</i>	4700.00
<i>Recursos de software</i>	0.00
<i>Gastos generales</i>	432.72
<i>Beneficio industrial</i>	366.85
<i>IVA</i>	1490.62
TOTAL	7098.19

Tabla 7: Presupuesto total

Los gastos generales se han considerado un 9% de los gastos en recursos, de la misma manera los beneficios industriales se han estimado en un 7% de los gastos incurridos.

GLOSARIO

Ajax. Proviene del inglés, Asynchronous JavaScript and XML. Tecnología utilizada la realizar la comunicación con la base de datos.

API. Proviene del inglés, Application Programming Interface. Conjunto de métodos de comunicación entre varios componentes a nivel de software.

CSS. Proviene del inglés, Cascade Style Sheets. Lenguaje de hojas de estilo utilizado para el diseño de los elementos especificados en el código HTML.

Framework.

HTML. Proviene del inglés, HyperText Markup Language. Lenguaje utilizado para la programación de páginas web, definiendo la estructura básica de elementos que compondrán la página en sí.

JSON. Proviene del inglés, JavaScript Object Notation. Formato de texto ligero diseñado principalmente para el intercambio de datos.

Mock-up. Prototipo o modelo utilizado para fines experimentales o de demostración.

Plugin. Aplicación que añade funcionalidad adicional o una nueva característica a un software existente.

Script. Consiste en un programa usualmente simple, que por lo general se almacena en un archivo de texto plano y son interpretados, es decir necesitan de otro software para realizar una funcionalidad.

SQL. Proviene del inglés, Structured Query Language. Lenguaje de programación utilizado para el diseño y manejo de bases de datos.

URL. Proviene del inglés, Uniform Resource Locator. En castellano localizador de recursos uniforme, secuencia de caracteres utilizada para localizar recursos a través de internet.

AUTORÍA DE IMÁGENES

Los iconos utilizados han sido desarrollados y pertenecen a la librería Font Awesome, pero son de uso libre.

Las imágenes utilizadas han sido diseñadas y el copyright pertenece a Sooodesign, pero son de uso libre.

Los diferentes diagramas han sido creados mediante la herramienta online *draw.io*.

La ilustración de casos de uso fue realizada mediante la herramienta online *Createely*.

BIBLIOGRAFÍA

- Luijbregts, B. (2018). HTML, CSS, and JavaScript: The Big Picture. Obtenido de <https://www.pluralsight.com/courses/html-css-javascript-big-picture>
- Zamoyta, M. (2018) JavaScript: Getting Started. Obtenido de <https://www.pluralsight.com/courses/javascript-getting-started>
- Zamoyta, M. (2018) JavaScript Fundamentals. Obtenido de <https://www.pluralsight.com/courses/javascript-fundamentals>
- Honeycutt, M. (2017) Building Mobile Apps with Visual Studio Tools for Apache Cordova. Obtenido de <https://www.pluralsight.com/courses/building-mobile-apps-visual-studio-apache-cordova>
- Allen, S. (2014) AngularJS: Get Started. Obtenido de <https://www.pluralsight.com/courses/angularjs-get-started>
- Allen, S. (2015) AngularJS Fundamentals. Obtenido de <https://www.pluralsight.com/courses/angularjs-fundamentals>
- Wildermuth, S. (2018) Building a Web App with ASP.NET Core, MVC, Entity Framework Core, Bootstrap, and Angular. Obtenido de <https://www.pluralsight.com/courses/aspnetcore-mvc-efcore-bootstrap-angular-web>
- Buna, S. (2018) Node.js: Getting Started. Obtenido de <https://www.pluralsight.com/courses/nodejs-getting-started>
- Wahlin D. (2011) jQuery Fundamentals Obtenido de <https://www.pluralsight.com/courses/jquery-fundamentals>
- Intro to HTML and CSS. Obtenido de: <https://classroom.udacity.com/courses/ud001>
- The Complete JavaScript Course For Web Development Beginners. Obtenido de: <https://www.udemy.com/javascriptcourse>
- PhoneGap - LearnToProgram: Become a Web or Mobile Developer. Obtenido de <https://www.youtube.com/watch?v=BbWNmZbLkhg&list=PLAgyIfU8wrtv2hcWbBAVKGYewrEN7vpPz>
- Getting Started with Apache Cordova Mobile App Development - Microsoft Visual Studio. Obtenido de : https://www.youtube.com/watch?v=_k9ZGXvV2PY&list=PLReL099Y5nRd9BNsMZwXvTDeqnfRMiGJy
- Documentation Cordova Apache. Obtenido de: <https://cordova.apache.org/docs/en/latest/>
- Documentation Ionic. Obtenido de: <https://ionicframework.com/docs>
- Documentation SurveyJS. Obtenido de: <https://surveyjs.io/Documentation/Library/>
- Documentation Bootstrap. Obtenido de: <https://getbootstrap.com/docs/4.0/getting-started/introduction/>
- Documentation AngularJS. Obtenido de: <https://docs.angularjs.org/api>
- Documentation jQuery. Obtenido de: <https://api.jquery.com/>

Documentation Font Awesome. Obtenido de: <https://fontawesome.com/how-to-use/on-the-web/setup/getting-started>

Documentation OnsenUI. Obtenido de: <https://onsen.io/v2/guide/>

Documentation Plugun-Local-Notification. Obtenido de:
<https://github.com/katzer/cordova-plugin-local-notifications/wiki>

Documentation Bootstrap Material Design. Obtenido de:
<https://fezvrasta.github.io/bootstrap-material-design/docs/4.0/getting-started/introduction/>

References HTML. Obtenido de:
<https://developer.mozilla.org/en-US/docs/Web/HTML/Element>

Refences HTML-Tags. Obtenido de: <https://www.w3schools.com/tags/default.asp>

References JavaScript. Obtenido de: <https://www.w3schools.com/jsref/default.asp>

Learn about mobile development with JavaScript in Visual Studio 2017. Obtenido de:
<https://docs.microsoft.com/en-us/visualstudio/cross-platform/tools-for-cordova/?view=toolsforcordova-2017> -

Several posts. Obtenido de: <https://stackoverflow.com/>

Petkobski, F. (2015) Apache Cordova Tutorial: Developing Mobile Applications with Cordova. Obtenido de: <https://www.toptal.com/mobile/developing-mobile-applications-with-apache-cordova>

Palme, E. (2018) How to wrap an Angular app with Apache Cordova. Obtenido de:
<https://medium.com/@EliaPalme/how-to-wrap-an-angular-app-with-apache-cordova-909024a25d79>

Ganushchak, S. (2017) How to Create App Design That Works. Obtenido de:
<https://yalantis.com/blog/how-to-create-app-design-that-works/>

Ziflaj, A. (2014) Native vs Hybrid App Development. Obtenido de:
<https://www.sitepoint.com/native-vs-hybrid-app-development/>

Yong Mook K. (2013) How to access JSON object in JavaScript. Obtenido de:
<https://www.mkyong.com/javascript/how-to-access-json-object-in-javascript/>

LocalStorage, sessionStorage. Obtenido de: <https://javascript.info/localstorage>

Boudreaux R. (2013) CSS specificity hierarchy: what you need to know. Obtenido de:
<https://www.techrepublic.com/blog/web-designer/css-specificity-hierarchy-what-you-need-to-know/>

Darlington, I. (2016) The 3 Types of Relationships in Database Design. Obtenido de:
<https://database.guide/the-3-types-of-relationships-in-database-design/>

Use Case Diagram. Obtenido de: <https://www.smartdraw.com/use-case-diagram/>

Rivera Casado, J. (2013) Hospital Clínico San Carlos 225 años enseñando medicina. Obtenido de: <https://dialnet.unirioja.es/servlet/articulo?codigo=4400637>

Información sobre el Hospital Clínico San Carlos. Obtenido de:
http://www.madrid.org/cs/Satellite?cid=1191579462083&language=es&pagename=HospitalClinicoSanCarlos%2FPagina%2FHCLN_contenidoFinal

Bibliografía

Hospital Clínico San Carlos. Obtenido de:

https://es.wikipedia.org/wiki/Hospital_Cl%C3%ADnico_San_Carlos

The Apache Software Foundation. Obtenido de:

https://en.wikipedia.org/wiki/The_Apache_Software_Foundation

General Data Protection Regulation. Obtenido de:

https://en.wikipedia.org/wiki/General_Data_Protection_Regulation

Github. Obtenido de: <https://en.wikipedia.org/wiki/GitHub>

MariaDB. Obtenido de: <https://en.wikipedia.org/wiki/MariaDB>

JSON. Obtenido de: <https://en.wikipedia.org/wiki/JSON>

JavaScript. Obtenido de: <https://en.wikipedia.org/wiki/JavaScript>

Cascading Style Sheets. Obtenido de:

https://en.wikipedia.org/wiki/Cascading_Style_Sheets

MariaDB. Obtenido de: <https://en.wikipedia.org/wiki/MariaDB>

Ecma International. Obtenido de: https://en.wikipedia.org/wiki/Ecma_International

HTML. Obtenido de: <https://en.wikipedia.org/wiki/HTML>

One-to-many (data model). Obtenido de: [https://en.wikipedia.org/wiki/One-to-many_\(data_model\)](https://en.wikipedia.org/wiki/One-to-many_(data_model))

jQuery. Obtenido de: <https://es.wikipedia.org/wiki/JQuery>

ANEXO

Anexo I: Código desarrollado

Con el objetivo de facilitar su difusión, el código desarrollado se encuentra en los siguientes repositorios:

- Aplicación: <https://github.com/mikelon797/TFG-San-Carlos>
- Base de datos: <https://github.com/mikelon797/BBDD-TFG-San-Carlos>



POLITÉCNICA

**ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES
UNIVERSIDAD POLITÉCNICA DE MADRID**

José Gutiérrez Abascal, 2. 28006 Madrid
Tel.: 91 336 3060
info.industriales@upm.es

www.industriales.upm.es