Universidad de Valladolid

E. T. S. de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Mención en Tecnologías de la Información

Estudio y desarrollo de un gestor de incidencias portable: UVaHelpDesk

Autor: D. Marcos García Alonso

Tutor: Dr. Arturo González Escribano

AGRADECIMIENTOS

A mi familia, que ha hecho posible que este trabajo haya sido una realidad con su sacrificio, siempre estaré en deuda con vosotros.

A mis amig@s, a los nuevos y a los viejos, a los que no han dudado de mí y me han regalado desinteresadamente su apoyo y consejo.

A mi tutor Arturo, por haber depositado su confianza y sabiduría en mí.

A mi compañero Álvaro, un ejemplo de trabajo, superación y compañerismo. Mi más sincera admiración.

RESUMEN

Ante la necesidad de muchas organizaciones de disponer de elementos de las IT y, en consecuencia, de personal que se encargue de su gestión y administración; así como de la resolución de las diferentes peticiones incidencias y problemas que puedan surgir eventualmente; se implementan los sistemas HelpDesk que se encargarán de facilitar el proceso de contacto entre los dos grupos de personas identificados: los usuarios y los equipos de soporte. Estos sistemas permiten crear las peticiones de servicio de forma que puedan llevar asociado un seguimiento, un control de calidad del servicio y de las estadísticas de todo el sistema en su totalidad.

Este trabajo abordará el análisis de las características de una serie de herramientas HelpDesk disponibles en el mercado e intentará dar solución a alguna de las carencias que estas puedan presentar con un producto destinado a un público no especializado.

Como resultado de este proyecto se ha logrado desarrollar una solución HelpDesk ligera, intuitiva, escalable y bien documentada y que soluciona el problema de portabilidad a diferentes dispositivos, incluyendo dispositivos móviles, satisfactoriamente.

ABSTRACT

Given the need of many Organizations of having IT elements and, consequently, counting with staff who take care of their management and administration; as well as the resolution of the different kind of requests or problems that could occur eventually; HelpDesk systems are implemented to ease the process of contact between these two groups of people identified: users and support teams. These systems allow the creation of service requests in a way that a tracing of them can be established along with the whole system service quality and statistics control.

This work will approach the analysis of some existing HelpDesk tools features available in the market and it will try to give a solution for some of the lacks that these tools may present with a product intended for non-specialized public.

As result of this project it has been achieved the developing of a light, intuitive HelpDesk solution, and solving the problem of portability between different kind of devices, including mobile devices, satisfactorily.

ÍNDICE DE CONTENIDO

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

1 BLOQUE I:
INTRODUCCIÓN

1.1 CONTEXTO

Este Trabajo de fin de Grado abordará el proceso de desarrollo de un producto Software dedicado a la gestión de las incidencias, peticiones, preguntas o problemas que se puedan dar lugar en una organización cualquiera generalmente en el ámbito de las IT (tecnologías de la información). Estos sistemas permiten poner en contacto a las personas a las que les surge algún tipo de problema con un grupo de técnicos especializados. Por medio del sistema se podrá realizar el proceso completo de la resolución de la incidencia, desde su notificación, hasta su resolución, pasando por su seguimiento. Estas aplicaciones son comúnmente denominadas *HelpDesk* o aplicaciones de *Ticketing.* Por ejemplo, un trabajador de una empresa necesita instalar un programa en su estación de trabajo, pero no tiene permisos para poder hacerlo desde su cuenta de usuario. Este trabajador notificará la necesidad de esta instalación a través de un HelpDesk. Inmediatamente aparecerá como una tarea en el HelpDesk a la persona encargada de este tipo de trabajo. Una vez instalado el programa remotamente, se cerrará la petición de servicio y se enviará un email automático al trabajador, se le dará la oportunidad de rellenar una encuesta, etc… Una serie de funciones que dependerán del sistema utilizado o sus opciones de configuración.

Estas aplicaciones normalmente giran alrededor del concepto de *ticket*: Como ticket se entiende la solicitud de servicio que se produce entre dos personas: una de ellas solicita a la otra la resolución de algún tipo de problema, la respuesta a una pregunta, la notificación de una incidencia… Por otra parte, habrá otra persona encargada de solucionar dicho problema y notificarlo, así como una tercera persona encargada de gestionar y distribuir las incidencias entre los diferentes técnicos. Esto hace que sea necesario hacer una distinción de las funcionalidades que se ofrecen dependiendo del rol que el usuario esté tomando en el momento de su utilización. Son, además, herramientas muy a menudo utilizadas en compañías medianas y grandes, sobre todo en el ámbito de las tecnologías de la información. En un mundo cada vez más avanzado tecnológicamente, el disponer de elementos del mundo de las IT es prácticamente una obligación en cualquier compañía, ya sea pública o privada. Por tanto, es necesario por parte de las organizaciones contar con un soporte de personal informático centralizado para llevar a cabo las tareas de administración y gestión de los recursos y resolución de problemas e incidencias con unas condiciones aceptables de calidad y seguridad. De esta forma se deriva de una forma eficaz el trabajo a los diferentes equipos de IT responsables de determinados centros/áreas, que además estarán especializados en los problemas recurrentes que puedan surgir en una determinada materia (redes, software, administración de cuentas de usuarios, servicio de soporte…).

Las herramientas de HelpDesk o Ticketing implementan una forma de enviar y realizar un seguimiento de las incidencias, solicitudes o problemas de forma automatizada y tener un orden a la hora de solucionarlas facilitando las metodologías de resolución por parte de los técnicos a la hora de resolver las solicitudes. Permiten realizar un seguimiento y obtener métricas que permitan marcar objetivos con la finalidad de mejorar el servicio de soporte al usuario final. También se suelen incorporar formas de valorar la satisfacción del cliente con el servicio ofrecido en forma de encuestas al darse por resuelto un ticket. Esto permitirá hacer futuras valoraciones o evaluaciones al personal técnico y a la calidad del servicio ofrecido.

Suelen también tener implementado algún tipo de SLA [1] (Service Level Agreement – Acuerdo con el Nivel de Servicio) con la finalidad de marcar objetivos de calidad a los técnicos en cuanto a factores como el tiempo de resolución, o la calidad de la atención al cliente. Los sistemas HelpDesk suelen ir acompañados además de gráficos y estadísticas para ofrecer un control más enriquecido del sistema de gestión de incidencias por parte de administradores o mánagers.

1.2 ESTUDIO DE APLICACIONES SIMILARES

Se ha hecho un estudio breve de una serie de aplicaciones similares de las que mayor acogimiento tienen en el mercado, para así analizar sus flaquezas y elaborar un producto distinto y diferenciado, que aporte algo que el resto de aplicaciones no hacen, buscando un nicho de mejora en un ámbito que ya está bastante abordado. Se han estudiado tanto herramientas comerciales con licencia, como herramientas open source para ayudar a tomar la decisión de qué tipo de licencia se empleará en este trabajo y ver cuáles son las diferencias comunes entre ambas.

Estas aplicaciones son ejemplos representativos de las diferentes clases de soluciones respecto a una serie de características relevantes que se han extraído y documentado en la Tabla 1.

Las aplicaciones elegidas han sido:

- ZenDesk [2]: https://www.zendesk.com/
- osTicket [3]: Self Hosted, es necesario descargar el código y disponer de un servidor web. http://osticket.com/download
- SupportSystem [4]: https://supportsystem.com/
- User Voice [5]: https://www.uservoice.com/
- Request Tracker [6]: https://bestpractical.com/rt-and-rti

De estas soluciones, dos de ellas son comerciales: ZenDesk y UserVoice. Otra open source: Request Tracker. Por último, osTicket, que presenta ambas posibilidades, una versión self-hosted y open source en la cual es necesario descargar el código y alojarlo en un servidor web, y una versión cloud-hosted denominada SupportSystem, que es alojada en la nube, pero esta vez con licencia comercial.

En las aplicaciones estudiadas destaca el hecho de que en ninguna de las que tienen licencia pública se ha implementado de forma efectiva su portabilidad a dispositivos móviles. La única que dispone de una aplicación móvil específica es UserVoice (Figura 1), pero se trata de una solución con licencia comercial. Así mismo, ZenDesk también dispone de tecnologías que permiten que el contenido se adapte al tamaño de la pantalla del dispositivo, pero también se trata de una solución comercial. Dado que en ninguna de las aplicaciones estudiadas con licencia pública disponen de una forma eficaz de mostrar su contenido en dispositivos móviles, este será un aspecto importante a tener en cuenta a la hora de elaborar la solución de este trabajo.

En todas las aplicaciones estudiadas existe una distinción necesaria de usuarios en base a su rol dentro del sistema. Los tipos de usuario imprescindibles en todas ellas son el usuario final y el administrador o agente. El usuario final será el usuario más común, el que notificará incidencias, problemas o peticiones.

En cuanto a la integración con redes sociales (RRSS), la única solución que dispone de alguna funcionalidad relacionada con ellas es ZenDesk, que permite hacer login en la aplicación por medio de Google+. Esto también puede ser un factor interesante que explotar en un posible trabajo futuro, dado que para el usuario suele ser más cómodo hacer login mediante una cuenta ya existente de una red social en lugar de tener que abrir una nueva cuenta de usuario con todo lo que ello conlleva: recordar una nueva contraseña, introducir los datos de registro, etc… Por otra parte, existen usuarios que pueden desconfiar de las redes sociales y su forma de compartir los datos y directamente no las utilicen, por lo que se ofrecen ambas opciones de login. Esta es una funcionalidad que puede ser interesante y se estudiará su inclusión en el producto en función del tiempo y los recursos disponibles.

Las aplicaciones HelpDesk suelen categorizar los tickets dependiendo de su naturaleza, un ticket puede ser creado como “Pregunta”, “Incidencia”, “Petición”, etc… Esta categorización puede servir para clasificar de alguna manera los tickets o aplicarles diferentes políticas de SLA en función de su tipo.

Por último, se han añadido en la Tabla 1 una serie de comentarios u observaciones con algunos detalles que llaman la atención o pueden resultar interesantes para el producto frente al resto de soluciones.

Software	Licencia	Tipos usuarios	Tipos tickets	Código Adaptable	Integración con RRSS	Versión Movil	Funcionalidades adicionales / Comentarios
ZenDesk	Comercial: Desde 9€ al mes (+5€ por agente)	- Usuario final - Agente - Administrador	- Pregunta - Incidente - Problema - Tarea	No (cloud hosted)	Google+	Se adapta pero los tamaños quedan muy pequeños	- Macros (automatización de tareas) - Tiempos de respuesta mejorables
osTicket	GNU (self hosted)	- Usuario Final - Agente - Administrador	- Ticket (sin distinción de tipo)	Sí (PHP)	No	No	
SupportSystem (osTicket cloud hosted)	Comercial: Desde 9€ por agente al mes	- Usuario final - Agente - Administrador	- Feedback - General Inquiry - Report a problem - Report a Problem / Access Issue	No (cloud hosted)	No	No	
UserVoice	Comercial: Desde 45$ por agente al mes	- Usuario final - Agente	- Support Request - Miscellaneous	No (cloud hosted)	No	- App móvil	- Widgets para sitios externos. Feedback. - Poco intuitiva la gestión de usuarios - Tiempos de respuesta mejorables - Importar contenido de otros servicios
Request Tracker	GNU	- Usuario final - Administrador	- Ticket (sin distinción de tipo)	Si (Perl)	No	No	- Solo disponible para Linux

Tabla 1: Comparativa entre aplicaciones HelpDesk similares

De la clasificación presentada en la Tabla 1 se puede inferir que osTicket es la solución que presenta características más interesantes para este estudio, a la hora de hacer una comparativa o análisis más profundo de una aplicación de este tipo por el siguiente motivo: se trata de una aplicación con licencia pública GNU, con lo que también se podrá hacer una comparativa en base al diseño arquitectónico y los patrones o frameworks empleados, ya que se dispone del código fuente en su totalidad. Por este motivo la comparativa se centrará en osTicket y se estudiarán puntualmente otras aplicaciones con características de las que osTicket carece.

A continuación, se hará un breve análisis de la aplicación osTicket y sus elementos más destacados y fundamentales, de los que se hará uso cada vez que se utilice la aplicación. Esto son, los formularios que permiten cargar información en la base de datos, los menús que permiten navegar a través de todas las páginas, y las tablas que facilitan la visualización de listados de tickets.

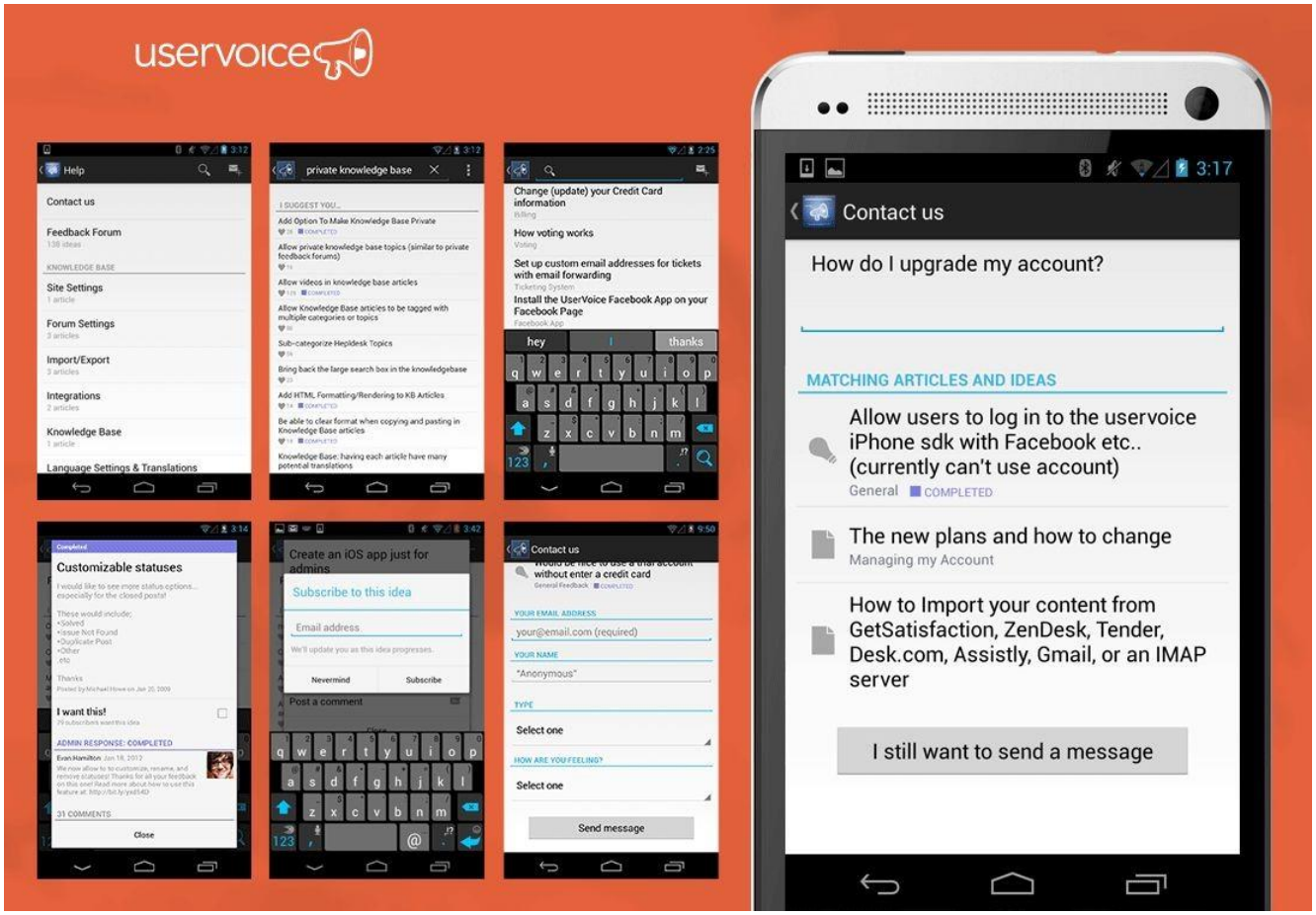


Figura 1: App móvil de UserVoice

1.2.1 Formularios de osTicket

La interfaz de usuario de osTicket es algo primitiva, con esto se quiere decir que no hace uso de las últimas tendencias o tecnologías a la hora de elaborar este tipo de interfaces de usuario en páginas webs o aplicaciones. Se limita a cumplir con las funcionalidades sin tener muy en cuenta la interacción entre la persona y la computadora, un aspecto cuidado y atractivo o el Diseño Centrado en el Usuario (DCU). La interfaz siempre es la misma sin importar las dimensiones del dispositivo sobre el cual se está ejecutando, con un ancho fijo y con un estilo simple. Esto empieza a ser un problema importante a la hora de emplearlo en un dispositivo móvil (Figura 3): el ancho de la página no se adapta y es necesario o bien verlo excesivamente pequeño, o bien ampliar la página e ir desplazándose a lo largo de la misma, siendo incómodo e ineficaz su uso. Mientras que en el navegador del ordenador (Figura 4) se puede ver de una forma más adaptada a la pantalla, pero aun así hay que hacer scrolling para recorrer el formulario entero y rellenar demasiados campos para crear una incidencia. Se busca que el ticket sea un elemento que se vea de un golpe visual rápidamente, que facilite la memoria fotográfica y sea rápido de crear; el hecho de tener que hacer scrolling desarticula la homogeneidad del ticket a nivel visual y resulta algo desconcertante al usuario final.

Si es necesario aportar información de ayuda a la hora de rellenar algún campo se hará de forma elegante y sin contaminar la interfaz de usuario, es decir: sin información que, pese a poder servir de ayuda, solamente sea necesaria mostrarla en determinadas ocasiones o incluso una única vez. Como se ve en el ejemplo de la Figura 2: ¿Es realmente necesario informar al usuario todas las veces que va a abrir un ticket de que la fecha está

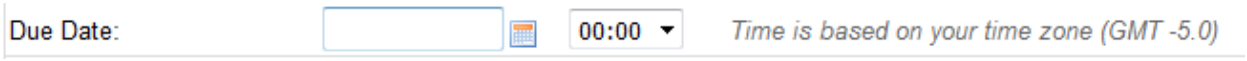


Figura 2: Introducción de fecha en osTicket

basada en su zona horaria? El hecho de introducir un campo sencillo como puede ser una fecha utiliza hasta 5 elementos visuales en este caso: una etiqueta, un campo de texto, un icono, un desplegable y un campo informativo.

Se debería optar en la medida de lo posible por campos autoexplicativos y no muy recargados de información, empleando efectos, colores y fuentes para resaltar campos guiando al usuario de una forma natural, minimizando el periodo y coste de reflexión por su parte.

1.2.2 Menús de osTicket

Considerando los menús como las zonas de accesos rápidos a los elementos de la aplicación, estos deberían ser visibles desde cualquier página para la correcta navegación a lo largo de ella. Esto puede suponer un problema si no se ha previsto una forma de adaptarlo a los dispositivos móviles, ya que las limitaciones que ofrecen las pantallas de estos en cuanto a dimensiones pueden dar lugar a complicaciones a la hora de organizar los elementos.

Como se puede ver en la Figura 3 y en la Figura 4, los menús de osTicket permanecen impasibles a pesar de cambiar el tamaño de la pantalla, tampoco es algo muy negativo ya que consta de un menú más bien sencillo, sin muchos elementos anidados en él, pero no deja de ser un inconveniente a la hora de utilizarlo y este hecho denota, además, que no ha habido una intención de construir un producto adaptable a este tipo de dispositivos.

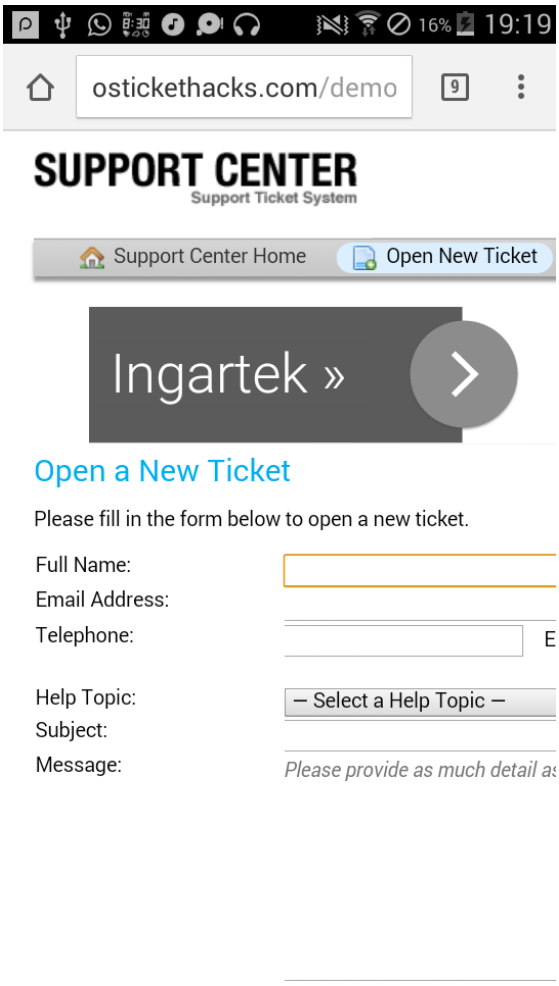


Figura 3: Interfaz osTicket móvil

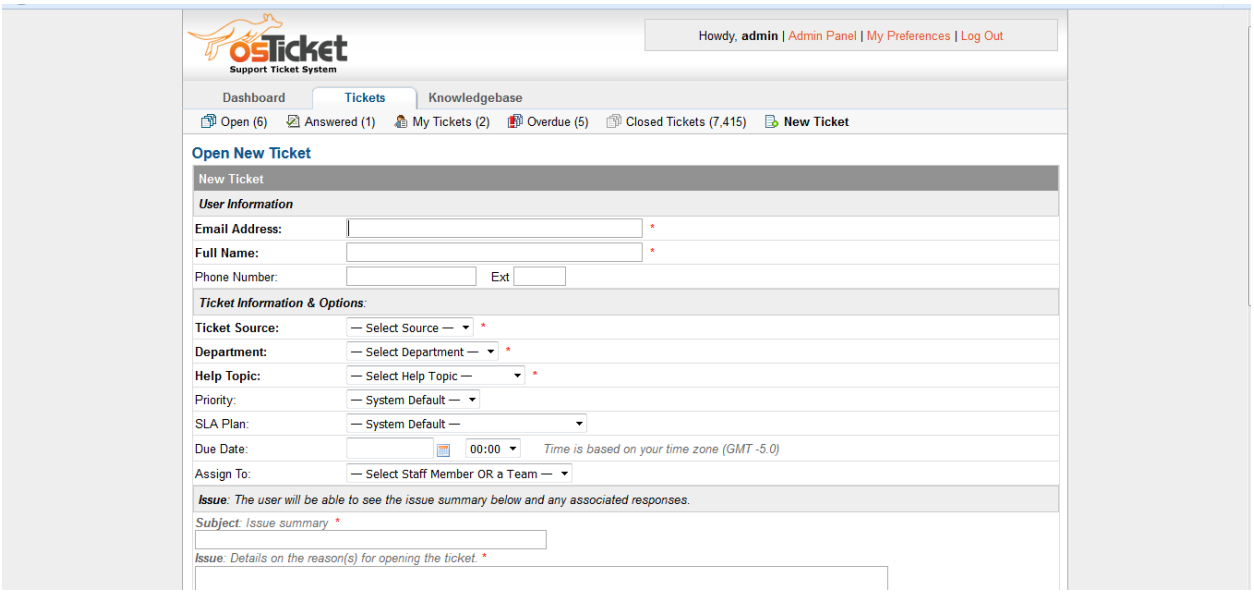


Figura 4: Interfaz osTicket navegador

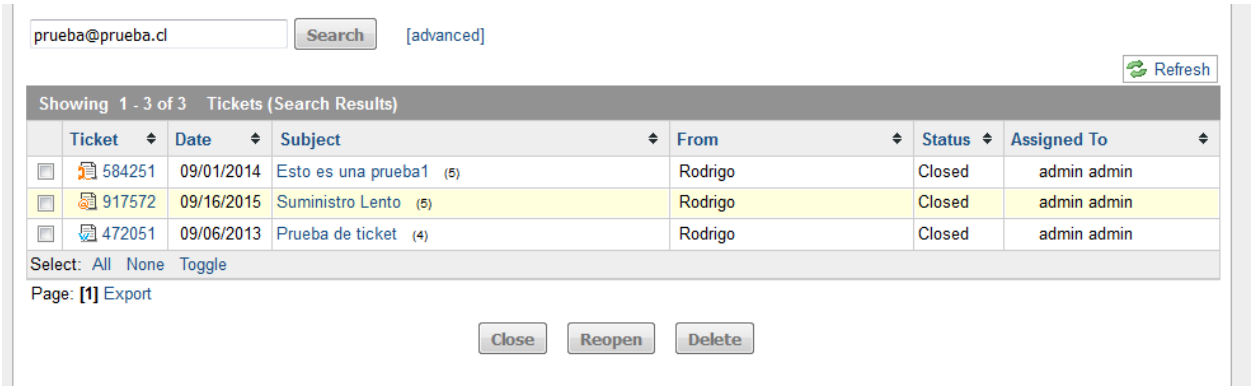


Figura 5: Tabla en osTicket

1.2.3 Tablas de osTicket

Otro elemento muy recurrente en las aplicaciones HelpDesk son las tablas, permitiéndose mostrar diferentes listados de tickets a los que se aplican unos determinados filtros: listados de tickets abiertos, listados de tickets asignados a un técnico en concreto, listados de tickets de una determinada fecha, etc…

osTicket proporciona estos listados en forma de tablas de una forma clásica (Figura 5): con la opción de ordenar cada una de las columnas y con un cuadro de búsqueda "Search" para filtrar las filas que contengan el texto introducido en alguno de sus campos. Cada una de estas búsquedas requiere una petición síncrona al servidor, con lo que es imprescindible una recarga de la página al completo.

Sería una opción interesante romper con este mecanismo de comunicación síncrona con el servidor por medio de técnicas de comunicación asíncronas que permitan que el intercambio de información sencilla no requiera una recarga de la página al completo. De esta forma se aportará mayor fluidez en la utilización del programa y mayor sensación de dinamismo para el usuario.

1.3 Planteamiento de Solución

Tras analizar con un detalle suficiente una serie de herramientas HelpDesk disponibles en el mercado, tanto comerciales como de código abierto, se ha notado que se tratan de herramientas muy orientadas al entorno de trabajo y que dejan de lado aspectos que podrían llegar a ser muy interesantes en otro tipo de ámbitos como puede ser el de la portabilidad entre otros tipos de dispositivos, como smartphones y tablets, o la sencillez de uso.

Estos dispositivos tienen unas características un tanto diferentes a un PC o una estación de trabajo, como pueden ser las especificaciones técnicas del hardware que tienen equipado (por lo normal el rendimiento que ofrecen los dispositivos móviles es menor), así como el tamaño, resolución, gama de colores que estos presentan en sus pantallas... Se quiere dar cobertura a estas limitaciones técnicas manteniendo la compatibilidad con los dispositivos tradicionales y dar la posibilidad de utilizar esta herramienta en un entorno que no sea el puramente laboral.

El trabajo se enfocará mucho en la simpleza, el minimalismo y la intuición de uso por encima de otros aspectos como pueden ser funcionalidades muy exclusivas que no se usen en gran parte de las aplicaciones que pueda tener una herramienta de HelpDesk o Ticketing.

Se trata ante todo de dar la oportunidad a los usuarios de notificar una incidencia, hacer una pregunta, poner en contacto a una persona que necesita algo y a otra que lo sabe resolver de una forma rápida, sin dar rodeos a lo largo de la herramienta, evitando ventanas emergentes y haciéndolo de la forma más cómoda y rápida posible para el usuario. Suele ser molesto para los usuarios el hecho de tener que notificar algo que funciona mal a través de algo que puede generar una mala experiencia, que es lento o de complicada utilización. Por ello se plantearán una serie de condiciones en forma de métricas concretas y medibles que se deben cumplir

numéricamente, siempre que sea posible, se conseguirá así una forma objetiva de evaluar si se ha conseguido la experiencia de usuario deseada. Se comentarán los requisitos generales que se deben cumplir y más adelante se asociarán estos requisitos con sus métricas correspondientes:

1.3.1 Sencillez

Se desea una aplicación que sea fácil de usar para un usuario que no ha recibido ningún tipo de formación en la misma. Se trata de facilitar su uso en entornos no especializados. Esto se logrará con campos autoexplicativos, que no confundan al usuario, claros, concisos y concretos, evitando la inclusión de la información menos relevante.

1.3.2 Aspecto elegante

Con el avance de las tecnologías se está tendiendo cada vez más a la construcción de herramientas que resulten atractivas a la vista, combinando elementos incluso artísticos con los puramente técnicos. Antaño este aspecto estaba muy descuidado, ya que la tendencia era centrarse exclusivamente en el aspecto técnico, dejando de lado la ergonomía, estética o el diseño. Esto es comprensible dado que las capacidades de cálculo eran mucho más limitadas que las disponibles actualmente en los dispositivos y no era funcional sacrificar la capacidad de cálculo en detrimento de un aspecto atractivo.

Esto ha cambiado mucho, y actualmente se da mucha importancia al aspecto visual, siendo cada vez más crítico el usuario (fundamentalmente el menos especializado) con este aspecto.

1.3.3 Tiempo de uso breve

Se trata de minimizar al máximo posible el tiempo de utilización para los usuarios finales. El tiempo de creación de un ticket, así como la búsqueda de un ticket abierto para comprobar el estado en el que se encuentra debe ser aceptable. Esto se logrará minimizando las recargas de página y el tiempo que emplea el usuario en discurrir y rellenar los formularios.

1.4 OBJETIVOS

En la siguiente lista se enumerarán los objetivos principales del trabajo de una forma lo más resumida posible:

- Se quiere una aplicación tipo HelpDesk accesible desde dispositivos de diferente naturaleza: Smartphones, tablets y ordenadores de escritorio y portátiles. El hecho de que la aplicación sea adaptable a dispositivos móviles no debe suponer un aumento notable de la complejidad en la implementación.
- La apertura de incidencias debe ser rápida y eficaz. Se debe poder realizar un seguimiento de las mismas.
- La aplicación debe tener al menos 3 tipos de usuarios con interfaces diferentes: cliente que notificará incidencias, técnico que las resolverá y mánager que asignará trabajos y controlará la calidad del servicio
- Se debe dar la oportunidad a ser empleada en diversos ámbitos, no solo el de IT.
- La arquitectura software estará basada en patrones de diseño y métodos comprobados y bien documentados que han resultado en productos de éxito a lo largo de la historia reciente de la Ingeniería del Software.

1.5 ESTRUCTURA DEL DOCUMENTO

Este documento está compuesto por una serie de bloques que corresponderán a cada una de las fases del desarrollo del producto en su totalidad tratando de reflejar todos los detalles importantes del mismo.

- **BLOQUE I: Introducción.**
  En el primer bloque se ha hecho un repaso general del trabajo a realizar incluyéndose un estudio de una serie de aplicaciones disponibles en el mercado tanto comerciales, como de código abierto, tratando de buscar un hueco que la aplicación que se elaborará pueda cubrir o mejorar. A partir de dicho estudio se plantea una propuesta de una solución cuyo desarrollo se detallará en los sucesivos bloques de esta memoria.
  Por último, este bloque enumera una serie de objetivos que deberán cumplirse satisfactoriamente al final del proceso de desarrollo.

- **BLOQUE II: Fundamentos teóricos.**
  En el segundo bloque se explicarán una serie de conceptos, herramientas, y tecnologías que son fundamentales a la hora de elaborar el producto final con una breve descripción de las misma y acompañados de una justificación que explica el porqué de su utilización frente a otras alternativas.

- **BLOQUE III: Planificación.**
  En el tercer bloque se expone de forma detallada el proceso de desarrollo que se ha de seguir en la elaboración de la solución. El proceso de desarrollo vendrá dividido en una serie de fases que se explicarán en este bloque.
  Se presentará una estimación de los costes de desarrollo teniendo en cuenta una serie de factores como el tiempo de desarrollo, y las herramientas empleadas.
  Por último, se hará un breve estudio de los riesgos que pueden surgir en el proceso, junto con los planes de contingencia previstos en caso de que dichos riesgos ocurran en alguna parte del proceso.

- **BLOQUE IV: Análisis de requisitos.**
  En el cuarto bloque se hará un análisis de los requisitos tanto funcionales como no funcionales que la aplicación debe cumplir. Se comenzará estableciendo una serie de métricas para poder evaluar en un futuro la cumplimentación de los requisitos no funcionales de una forma precisa y concreta.
  A continuación, se definirán en detalle los datos y entidades que la aplicación manejará, para así elaborar los requisitos funcionales.
  Por último, se desglosarán estas funcionalidades en una serie de casos de uso que servirán como guía a la hora de elaborar el diseño final de la solución.

- **BLOQUE V: Diseño.**
  En el quinto bloque se tomarán las decisiones de diseño que se consideren más adecuadas dadas las características analizadas anteriormente. Se atenderá a los aspectos de la organización del sistema, y el flujo de control, tomándose siempre como referencia documentos bibliográficos bien asentados y aceptados por la comunidad, y creándose así un diseño arquitectónico global de la solución.
  De esta definición de diseño arquitectónico se extraerán una serie de modelos estáticos del sistema en forma de modelos gráficos explicados que conformarán cada una de las piezas que compondrán el sistema en su totalidad; así como una serie de modelos dinámicos que indicarán cómo se comportan e interrelacionan estos elementos entre sí.

- **BLOQUE VI: Implementación.**
  En el sexto bloque se comentarán los aspectos de implementación que puedan resultar más particulares o llamativos dentro de esta solución.

- **BLOQUE VII: Pruebas y resultados.**
  En el séptimo bloque se establecerá un plan de pruebas para comprobar que el funcionamiento de la solución concuerda con los criterios establecidos en las primeras fases del desarrollo. Se diferenciarán dos grandes partes: las pruebas y los resultados. Como pruebas se entiende una serie de comprobaciones unitarias para verificar que los resultados obtenidos son los mismos que se establecieron en los casos de uso del Bloque IV, y de esta manera, verificar que los requisitos funcionales han sido cumplidos satisfactoriamente.
  Los resultados evaluarán las métricas definidas en el Bloque IV, para así evaluar de algún modo el grado de cumplimentación con los requisitos no funcionales definidos.

- **BLOQUE VIII: Conclusiones y trabajo futuro.**
  En el octavo y último bloque se enumerarán una serie de conclusiones íntimamente relacionadas con los objetivos definidos en el Bloque I y de qué manera se ha cumplido con los mismos al final del proceso de desarrollo.
  Se establecerán además unas líneas de posible trabajo futuro para una hipotética mejora o extensión del producto siguiendo la ruta marcada desde un principio.
  Por último y como colofón, se hará una valoración personal del autor de todo el proceso de desarrollo que abarca este trabajo.

2 BLOQUE II: FUNDAMENTOS TEÓRICOS

En esta sección se darán unas nociones puramente conceptuales para comprender los componentes y herramientas de apoyo que se emplearán a lo largo del desarrollo del trabajo. Estas definiciones irán acompañadas de la justificación de su utilización, y eventualmente de diferentes alternativas contempladas y descartadas.

2.1 HTTP

Hypertext Transfer Protocol o HTTP [7] [8] (en español protocolo de transferencia de hipertexto) es el protocolo de comunicación que permite las transferencias de información en la World Wide Web.

HTTP fue desarrollado por el World Wide Web Consortium (W3C) y la Internet Engineering Task Force (IEFT), colaboración que culminó en 1999 con la publicación de una serie de publicaciones Request For Comment (RFC), el más importante de ellos es el RFC 2616 que especifica la versión 1.1. HTTP define la sintaxis y la semántica que utilizan los elementos de software de la arquitectura web (clientes, servidores, proxys) para comunicarse. HTTP es un protocolo sin estado, es decir, no guarda ninguna información sobre conexiones anteriores. El desarrollo de aplicaciones web necesita frecuentemente mantener estado. Para esto se pueden usar las cookies, que es información que un servidor puede almacenar en el sistema cliente. Esto permite a las aplicaciones web instituir la noción de *sesión* (*session*), y también permite rastrear usuarios ya que las cookies pueden guardarse en el cliente por tiempo indeterminado.

Netscape Communications creó HTTPS en 1992 para su navegador. Estrictamente hablando, HTTPS no es un protocolo separado, pero refiere el uso del HTTP ordinario sobre una Capa de Conexión Segura cifrada Secure Sockets Layer (SSL) o una conexión con Seguridad de la Capa de Transporte (TLS).

Dado que estos protocolos constituyen el pilar fundamental de las comunicaciones entre los dispositivos que conforman *Internet*, se ha visto conveniente mencionarlo en este apartado ya que determinan características básicas de las aplicaciones web en general

2.2 HTML

HTML [9] o lenguaje de marcado de hipertexto, se ha convertido en el lenguaje de marca más importante a día de hoy, gracias al crecimiento de Internet, la necesidad de ofrecer servicios remotamente y, en definitiva, la necesidad de poder ofrecer fácilmente información de una forma elegante. HTML es un derivado de SGML y nació en 1991, gracias a Tim Berners-Lee (que trabajaba en el CERN), que ante la necesidad de compartir información entre científicos creó la primera definición del lenguaje. En enero de 1997 se publicó HTML3.2 como recomendación de la W3C, que fue la primera definición redactada exclusivamente por el consorcio, fuertemente influenciado por Netscape.

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

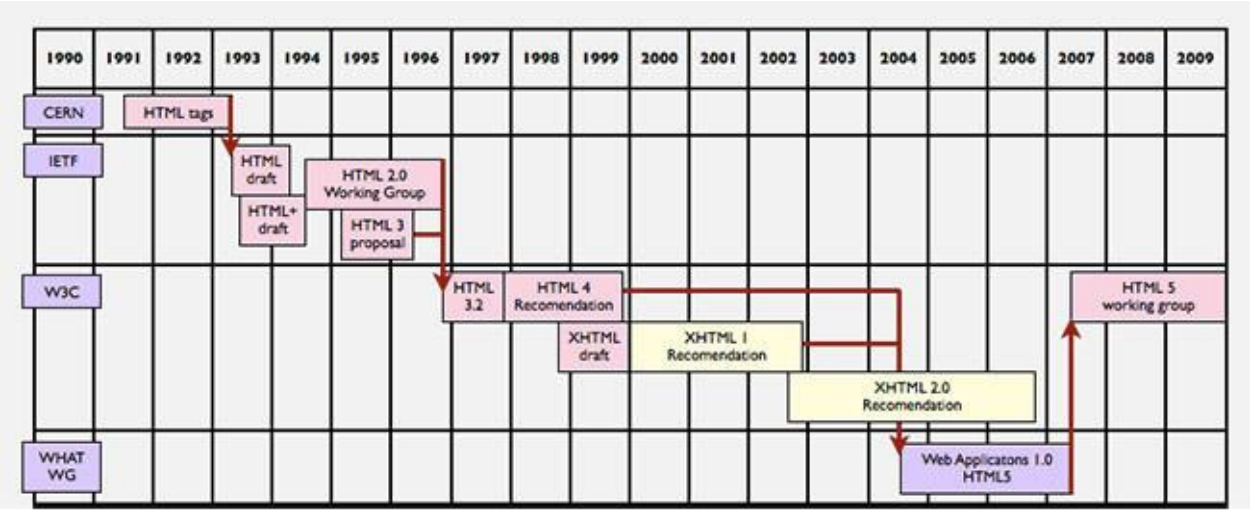


Figura 6: Evolución de HTML

HTML se escribe en forma de «etiquetas», rodeadas por corchetes angulares (<,>,/). HTML también puede describir, hasta un cierto punto, la apariencia de un documento (aunque se recomienda que esto se delegue en las hojas de estilo CSS, sección 2.3), y puede incluir o hacer referencia a un tipo de programa llamado script (como por ejemplo JavaScript, sección 2.9), el cual puede afectar el comportamiento de navegadores web y otros procesadores de HTML.

Seleccionando la opción *ver código fuente* o *inspeccionar elemento* en el navegador, se puede ver realmente la información que está recibiendo el navegador web y cómo la está interpretando. De esta forma, se abrirá el editor de texto configurado como predeterminado en el sistema con el código fuente HTML de la página que se esté viendo en ese momento en el explorador. Aparte de poder ver el código fuente HTML de una página web con las opciones antes descritas, Internet Explorer, Firefox y Google Chrome incorporan también unas herramientas conocidas como inspectores de página que se puede activar con F12.

Con estas herramientas es posible visualizar una página web y seleccionar dentro de ella un elemento concreto del cuál queremos conocer cuál es el código HTML con el que está implementado. Al hacer esto, el código se mostrará en un área especial dentro del navegador en el que el usuario podrá ver el código HTML en cuestión, además de las reglas CSS que aplican a ese código HTML en concreto. Este tipo de análisis resulta sumamente instructivo para aprender a programar en HTML. Para el navegador Firefox, además, existe como alternativa a la herramienta nativa el plugin Firebug, muy similar a la herramienta que Firefox incorpora por defecto. Esta herramienta será clave en el proceso de desarrollo dadas las posibilidades de depuración que esta ofrece.

HTML5 (HyperText Markup Language, versión 5) es la quinta revisión importante del lenguaje básico de la World Wide Web, HTML. HTML5 especifica dos variantes de sintaxis para HTML: una *clásica*, HTML (text/html), conocida como HTML5, y una variante XHTML conocida como sintaxis XHTML5 que deberá servirse con sintaxis XML (application/xhtml+xml). Esta es la primera vez que HTML y XHTML se han desarrollado en paralelo. La versión definitiva de la quinta revisión del estándar se publicó en octubre de 2014.

HTML5 será un elemento importante para el desarrollo de este trabajo ya que se hace uso de ello de una forma transparente a través del framework responsive *Bootstrap* (ver sección 2.8).

2.3 CSS

CSS [10] es un lenguaje de hojas de estilo creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas web con estilos estéticos y otras formas de aportar dinamismo.

```css
@font-face {
    font-family: 'FontAwesome';
    src: url('../fonts/fontawesome-webfont.eot?v=4.3.0');
    src: url('../fonts/fontawesome-webfont.eot?#iefix&v=4.3.0')
    font-weight: normal;
    font-style: normal
}

.fa {
    display: inline-block;
    font: normal normal normal 14px/1 FontAwesome;
    font-size: inherit;
    text-rendering: auto;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
    transform: translate(0, 0)
}

.fa-lg {
    font-size: 1.33333333em;
    line-height: .75em;
    vertical-align: -15%
```

Figura 7: Ejemplo de código CSS

Separar la definición de los contenidos y la definición de su aspecto presenta numerosas ventajas, ya que obliga a crear documentos HTML/XHTML bien definidos y con significado completo, también llamados "documentos semánticos". Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes.

Al crear una página web, se utiliza en primer lugar el lenguaje HTML/XHTML para marcar los contenidos, es decir, para designar la función de cada elemento dentro de la página: párrafo, titular, texto destacado, tabla, lista de elementos, etc.

Una vez creados los contenidos, se utiliza el lenguaje CSS (Figura 6) para definir el aspecto de cada elemento: color, tamaño y tipo de letra del texto, separación horizontal y vertical entre elementos, posición de cada elemento dentro de la página, etc.

El organismo W3C (World Wide Web Consortium), encargado de crear todos los estándares relacionados con la web, propuso la creación de un lenguaje de hojas de estilos específico para el lenguaje HTML y se presentaron nueve propuestas. Las dos propuestas que se tuvieron en cuenta fueron la CHSS (Cascading HTML Style Sheets) y la SSP (Stream-based Style Sheet Proposal).

La propuesta CHSS fue realizada por Håkon Wium Lie y SSP fue propuesto por Bert Bos. Entre finales de 1994 y 1995 Lie y Bos se unieron para definir un nuevo lenguaje que tomaba lo mejor de cada propuesta y lo llamaron CSS (Cascading Style Sheets). Este lenguaje es el que utiliza de forma generalizada hoy en día a la hora dew construir aplicaciones web. Las hojas de estilo son fundamentales para poder generar el aspecto visual de la aplicación tanto en navegadores tradicionales como en los que incluyen los dispositivos móviles.

2.4 PHP

PHP (acrónimo recursivo de PHP: Hypertext Preprocessor) [11] [12] es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML.

PHP está enfocado a la programación de scripts del lado del servidor, por lo que se puede hacer cualquier cosa que pueda hacer otro programa CGI, como recopilar datos de formularios, generar páginas con contenidos dinámicos, o enviar y recibir cookies.

Existen principalmente tres campos principales donde se usan scripts de PHP.

- Scripts del lado del servidor. Este es el campo más tradicional y el foco principal. Son necesarios tres elementos para que esto funcione. El analizador de PHP (módulo CGI o servidor), un servidor web y un navegador web. Es necesario ejecutar el servidor con una instalación de PHP conectada. Se puede acceder al resultado del programa de PHP con un navegador, viendo la página de PHP a través del servidor.
- Scripts desde la línea de comandos. Se puede crear un script de PHP y ejecutarlo sin necesidad de un servidor o navegador. Solamente es necesario el analizador de PHP para utilizarlo de esta manera. Este tipo de uso es ideal para scripts que se ejecuten con regularidad empleando cron (en Unix o Linux) o el Planificador de tareas (en Windows). Estos scripts también pueden usarse para tareas simples de procesamiento de texto.
- Escribir aplicaciones de escritorio. Probablemente PHP no sea el lenguaje más apropiado para crear aplicaciones de escritorio con una interfaz gráfica de usuario, pero si se conoce bien PHP, y se quisiera utilizar algunas características avanzadas de PHP en aplicaciones del lado del cliente, se puede utilizar PHP-GTK para escribir dichos programas. También es posible de esta manera escribir aplicaciones independientes de una plataforma. PHP-GTK es una extensión de PHP, no disponible en la distribución principal.

PHP puede emplearse en todos los sistemas operativos principales, incluyendo Linux, muchas variantes de Unix (incluyendo HP-UX, Solaris y OpenBSD), Microsoft Windows, Mac OS X, RISC OS y probablemente otros más. PHP admite la mayoría de servidores web de hoy en día, incluyendo Apache, IIS, y muchos otros. Esto incluye cualquier servidor web que pueda utilizar el binario de PHP FastCGI, como lighttpd y nginx. PHP funciona tanto como módulo como procesador de CGI.

De modo que, con PHP, se tiene la libertad de elegir el sistema operativo y el servidor web. Además, se tiene la posibilidad de utilizar programación por procedimientos o programación orientada a objetos (POO), o una mezcla de ambas.

Con PHP no se está limitado a generar HTML. Entre las capacidades de PHP se incluyen la creación de imágenes, ficheros PDF e incluso películas Flash (usando libswf y Ming) generadas sobre la marcha. También se puede generar fácilmente cualquier tipo de texto, como XHTML y cualquier otro tipo de fichero XML. PHP puede autogenerar estos ficheros y guardarlos en el sistema de ficheros en vez de imprimirlos en pantalla, creando una caché en el lado del servidor para contenido dinámico.

Una de las características más potentes y destacables de PHP es su soporte para un amplio abanico de bases de datos. Escribir una página web con acceso a una base de datos es increíblemente simple utilizando una de las extensiones específicas de bases de datos (p.ej., para MySQL), o utilizar una capa de abstracción como PDO, o conectarse a cualquier base de datos que admita el estándar de Conexión Abierta a Bases de Datos por medio de la extensión ODBC. Otras bases de datos podrían utilizar cURL o sockets, como lo hace CouchDB.

PHP también cuenta con soporte para comunicarse con otros servicios usando protocolos tales como LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM (en Windows) y muchos otros. También se pueden crear sockets de red puros e interactuar usando cualquier otro protocolo. PHP tiene soporte para el intercambio de datos complejos de WDDX entre virtualmente todos los lenguajes de programación web. Y hablando de interconexión, PHP tiene soporte para la instalación de objetos de Java y emplearlos de forma transparente como objetos de PHP.

Dados estos beneficios (flexibilidad del lenguaje, integración sencilla y eficaz con otros sistemas, sencillez de portabilidad y código abierto) y junto al gran acogimiento en el mercado que este lenguaje ofrece, se ha elegido esta solución frente a otras posibles soluciones que podrían también haber sido candidatas para esta función: como JSP, que no tiene tan buena integración con otros sistemas, o ASP muy dependiente de otros productos de Microsoft, como por ejemplo el IDE (Entorno de Desarrollo Integrado), lo cual incrementaría el coste final del desarrollo, al ser éstos productos de licencia comercial.

2.5 MySQL

MySQL [13] [14] es el servidor de bases de datos relacionales más popular, desarrollado y proporcionado por MySQL AB. MySQL AB es una empresa cuyo negocio consiste en proporcionar servicios en torno al servidor de bases de datos MySQL.

Las bases de datos de MySQL son relacionales. Una base de datos relacional almacena los datos en tablas separadas en lugar de poner todos los datos en un solo lugar. Esto agrega velocidad y flexibilidad. Las tablas son enlazadas al definir relaciones que hacen posible combinar datos de varias tablas cuando se necesitan consultar datos.

Comenzó siendo Open Source y las versiones de usuario lo siguen siendo, sin embargo, las versiones Enterprise incluyen productos o servicios adicionales tales como herramientas de monitorización y soporte oficial. En 2009 se creó un fork denominado MariaDB por algunos desarrolladores, incluidos algunos desarrolladores originales de MySQL, descontentos con el modelo de desarrollo y el hecho de que una misma empresa controle a la vez los productos MySQL y Oracle Database.

Se optará por esta solución debido a la sencillez y eficacia de su integración con PHP, tecnologías que muy a menudo vienen combinadas a la hora de construir aplicaciones web. También ha sido un factor fundamental para su elección el hecho de que la licencia sea gratuita a nivel de usuario.

2.6 Apache

El servidor HTTP Apache es un servidor web HTTP de código abierto, para plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP y la noción de sitio virtual.

Cuando comenzó su desarrollo en 1995 se basó inicialmente en código del popular NCSA HTTPd 1.3, pero más tarde fue reescrito por completo. El servidor Apache es desarrollado y mantenido por una comunidad de usuarios bajo la supervisión de la Apache Software Foundation dentro del proyecto HTTP Server (httpd).

Apache presenta entre otras características altamente configurables, bases de datos de autenticación y negociado de contenido, pero fue criticado por la falta de una interfaz gráfica que ayude en su configuración.

Apache tiene amplia aceptación en la red: desde 1996, Apache, es el servidor HTTP más usado. Jugó un papel fundamental en el desarrollo fundamental de la World Wide Web y alcanzó su máxima cuota de mercado en 2005 siendo el servidor empleado en el 70% de los sitios web en el mundo, sin embargo, ha sufrido un descenso en su cuota de mercado en los últimos años. (Estadísticas históricas y de uso diario proporcionadas por Netcraft). En 2009 se convirtió en el primer servidor web que alojó más de 100 millones de sitios web.

La mayoría de las vulnerabilidades de la seguridad descubiertas y resueltas tan sólo pueden ser aprovechadas por usuarios locales y no remotamente. Sin embargo, algunas se pueden accionar remotamente en ciertas situaciones, o explotar por los usuarios locales malévolos en las disposiciones de recibimiento compartidas que utilizan PHP como módulo de Apache.

Se elige Apache como servidor web dada su enorme difusión, el hecho de ser un sistema muy asentado en el mercado y su facilidad de integración con PHP y MySQL gracias a herramientas como XAMPP (desarrollado por la propia fundación Apache), en las que vienen estos tres elementos (Apache, MySQL y PHP) preconfigurados de forma que basta con una sencilla instalación para tener en funcionamiento la infraestructura más fundamental sobre la que la aplicación web funcionará. También se facilitará la configuración más básica de estos elementos a través de esta instalación.

2.7 AJAX

AJAX [15] [16], acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, mejorando la interactividad, velocidad y usabilidad en las aplicaciones.

Ajax es una tecnología asíncrona, en el sentido de que los datos adicionales se solicitan al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página. JavaScript es el lenguaje interpretado (scripting language) en el que normalmente se efectúan las funciones de llamada de Ajax mientras que el acceso a los datos se realiza mediante XMLHttpRequest, objeto disponible en los navegadores actuales. En cualquier caso, no es necesario que el contenido asíncrono esté formateado en XML.

Se decide emplear esta tecnología para aportar mayor fluidez en el uso de la aplicación, en casos donde los intercambios de información con el servidor son sencillos y no sea necesario un cambio de la vista a la hora de recibirlos. Por ejemplo, en los diferentes filtrados que puedan tener los listados en forma de tabla (ver sección 2.9.2).

2.8 FRAMEWORKS PARA CONTENIDO RESPONSIVE

Los Frameworks para contenido responsive tratan de solventar el problema de complejidad y tiempo que surge a la hora de elaborar las vistas de la parte Front End de una aplicación web. Con una serie de plantillas predefinidas se facilita mucho la construcción de la parte de cliente en una aplicación web por medio de tecnologías como HTML5, CSS3 y JavaScript. Los ejemplos más difundidos de estos Frameworks son [17] Bootstrap (el más utilizado), Foundation (principal competidor de Bootstrap), o soluciones como Material-UI o Materialize que aplican los principios de diseño material [18]. De entre todas ellas se elige Bootstrap por ser la que resulta más familiar para el desarrollador y la que más documentación puede tener dada su difusión, evitándose así inversiones en tiempo de aprendizaje y su coste asociado.

Bootstrap [17] [18] es un framework o conjunto de herramientas de software libre para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como, extensiones de JavaScript opcionales adicionales.

Bootstrap tiene un soporte relativamente incompleto para HTML5 y CSS 3, pero es compatible con la mayoría de los navegadores web. La información básica de compatibilidad de sitios web o aplicaciones está disponible para todos los dispositivos y navegadores. Existe un concepto de compatibilidad parcial que hace disponible la información básica de un sitio web para todos los dispositivos y navegadores.

Desde la versión 2.0 también soporta diseños sensibles. Esto significa que el diseño gráfico de la página se ajusta dinámicamente, tomando en cuenta las características del dispositivo usado (ordenador de sobremesa / portátil, tabletas, teléfonos móviles).

Bootstrap es de código abierto y está disponible en GitHub.

2.9 JAVASCRIPT

Javascript se ha utilizado en el desarrollo de este software para realizar funciones que se pueden realizar en la parte del cliente como validaciones y soporte para las llamadas AJAX.

Javascript es un lenguaje interpretado usado para múltiples propósitos, pero solo considerado como un complemento hasta ahora. Es un lenguaje pequeño y ligero, es menos útil como un lenguaje independiente, más bien está diseñado para una fácil incrustación en otros productos y aplicaciones. Se utiliza principalmente en su forma del lado del cliente (*client-side*), implementado como parte de un navegador web permitiendo mejoras

en la interfaz de usuario y páginas web dinámicas, aunque existe una forma de JavaScript del lado del servidor (Server-side JavaScript o SSJS).

Contrariamente a la falsa idea popular, JavaScript no es "Java interpretativo". En pocas palabras, JavaScript es un lenguaje de programación dinámico que soporta construcción de objetos basado en prototipos. La sintaxis básica es similar a Java y C++ con la intención de reducir el número de nuevos conceptos necesarios para aprender el lenguaje.

2.9.1 jQuery

JQuery [19] es una biblioteca de JavaScript, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web. JQuery es la biblioteca de JavaScript más utilizada.

JQuery es software libre y de código abierto, posee un doble licenciamiento bajo la Licencia MIT y la Licencia Pública General de GNU, permitiendo su uso en proyectos libres y privados. JQuery, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio.

JQuery consiste en un único fichero JavaScript que contiene las funcionalidades comunes de DOM, eventos, efectos y AJAX.

La característica principal de la biblioteca es que permite cambiar el contenido de una página web sin necesidad de recargarla, mediante la manipulación del árbol DOM y peticiones AJAX. Para ello, utiliza las funciones $() o jQuery().

2.9.2 jQuery DataTable

DataTables es un plugin para jQuery que nos permite darles dinamismo a nuestras tablas de una forma sencilla y aumentando la vistosidad de nuestras tablas. Con solo un par de líneas, se cambiará completamente la apariencia de nuestra tabla.

Con este plugin se conseguirán las siguientes acciones con las tablas:

- Ordenar por campos.
- Paginar resultados.
- Filtrar.
- Sistema de búsqueda.
- Obtención de datos de forma dinámica.

2.10 HERRAMIENTAS SOFTWARE UTILIZADAS

Se ha hecho un repaso a un número considerable de tecnologías que en conjunto constituyen las herramientas para el desarrollo del tipo de aplicación objetivo de este trabajo. Se ha hecho una breve evaluación entre tecnologías similares y se han descartado las que no podrían resultar tan útiles justificadamente.

El listado final de herramientas software a emplear es el que viene a continuación:

- **Sistema Operativo**: Windows 7 Professional
- **Productividad:** En esta categoría entrarán las aplicaciones que, a pesar de no producir documentación en sí, ni contribuir al desarrollo del producto estrictamente hablando, sí facilitan y agilizan ciertas tareas:
    - OneDrive: Almacenamiento de ficheros en la nube.
    - GreenShot: Tratamiento de capturas de pantalla.

- **Desarrollo y despliegue de infraestructura:** En esta categoría se encuentran las herramientas que han servido para la implementación de la solución, así como el despliegue de las infraestructuras sobre las que se sustenta:
    - Bootstrap: Framework para el desarrollo de sitios y aplicaciones web con diseño adaptativo. Contiene HTML, CSS y en ocasiones código JavaScript.
    - PHP 5.6.14: Código que el servidor ejecutará.
    - MySQL: Gestor de bases de datos.
    - Apache 2.4.17: Servidor web.
    - XAMPP 3.2.1: Herramienta para facilitar la integración y configuración de Apache, MySQL y PHP.
    - Eclipse Mars 4.5.1: Entorno de desarrollo utilizado para la implementación de la aplicación.
    - FileZilla: Intercambio de archivos por FTP para gestionar el sitio subido a un servidor remoto.
    - phpMyAdmin: Herramienta para la administración de bases de datos MySQL a través de páginas web.
    - Mozilla Firefox 49.0.1: Navegador web.
    - Firebug: Extensión de Firefox para facilitar la depuración y el desarrollo del sitio web.
- **Documentación:** En esta categoría se engloban las herramientas que se han empleado para generar la documentación del proceso de desarrollo del producto.
    - Microsoft Office 2016: Realización de la documentación: procesador de textos, tablas de cálculo, presentaciones de diapositivas.
    - Microsoft Project 2016: Herramienta para elaborar la planificación del trabajo.
    - Astah Professional 7: Herramienta para el modelado de diagramas UML.
    - Adobe Acrobat Reader DC 2015.020.20039.

3 BLOQUE III: PLANIFICACIÓN

3.1 PLANIFICACIÓN DEL TRABAJO

Se ha elaborado una planificación para el trabajo utilizando la herramienta de Microsoft: MS Project 2016. Con ella se obtienen una serie de datos y diagramas que servirán de soporte al plan de trabajo pensado para este proyecto.

Se tratará de seguir un plan de trabajo basado en una sencilla adaptación de los principios del Proceso Unificado [20], obteniéndose un desarrollo de software dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental.

- **Dirigido por casos de uso**: Los casos de uso han sido adoptados casi universalmente como el método de captura de requisitos de los sistemas software en general. Junto con los casos de uso se distinguirán una serie de actores por cada uno de los tipos de usuario identificados. Los actores utilizarán el sistema interactuando con los casos de uso. El tener una lista con una serie de casos de uso facilita guiar el proyecto a través de ellos. En la fase de pruebas (Bloque VII), también se facilitará elaborar un plan de pruebas para verificar que el sistema implementa satisfactoriamente estos casos de uso.
- **Centrado en la arquitectura**: Como arquitectura se entiende una visión global de todos los aspectos del desarrollo del producto. La arquitectura estará definida por una serie de modelos (modelo de casos de uso, modelo de análisis, modelo de diseño…) mediante el lenguaje de modelado UML. Esto facilitará el desarrollo, la evolución y la reutilización de los elementos del sistema.
- **Iterativo e incremental**: Con el objetivo de aportar equilibrio y coherencia entre los casos de uso y la arquitectura, se propone un plan de trabajo iterativo e incremental, con una serie de hitos que proporcionen al desarrollo una forma de evaluar si una fase del desarrollo ha terminado satisfactoriamente: se puede autorizar la continuación hacia la siguiente fase, o al contrario se debe hacer una revisión de ella.

De esta forma se definirá un detallado análisis del que se obtendrán un conjunto de casos de uso que dirigirán el proyecto en todas sus fases. Así mismo estos casos de uso pasarán por sucesivas revisiones para ir afinándolos y adecuándolos a la situación del proyecto. Estas revisiones a las que serán sometidas las fases del trabajo harán que este sea iterativo e incremental, afinándose su definición en cada iteración y, consecuentemente, incrementándose su complejidad en cada una de ellas.

Cada una de las fases llevará un proceso de documentación paralelo al trabajo a medida que se vaya elaborando. Al final habrá que unificar toda esta documentación en una memoria formal que englobe todo el proceso de desarrollo y fundamentación teórica. Los plazos y tiempos son aproximados y no tiene por qué cumplirse esta planificación a rajatabla, pero sí en la medida de lo posible. Estas desviaciones de la planificación se contemplarán en el apartado de riesgos (sección 3.4).

Las fases de las que consta el proyecto darán una serie de outputs, los cuales se resumen en la Tabla 2.

3.1.1 Fase de inicio

En esta fase se cubrirán las partes más prematuras del proyecto: una definición inicial al trabajo a realizar, junto con un estudio de aplicaciones de tipo HelpDesk con el objetivo de identificar aspectos únicos de este trabajo susceptibles de ser explotados en su favor frente a otras aplicaciones existentes.

Una vez se tenga una comparativa de un grupo de aplicaciones HelpDesk, se empezará a definir el alcance del producto. En la Figura 8 y en la Figura 9 se pueden ver las subfases de esta fase de inicio y el diagrama de Gantt resultante respectivamente.

Fase	Outputs
Inicio	- Visión general del sistema, requisitos mínimos - Objetivos - Primera aproximación al diseño arquitectónico, y subsistemas a utilizar - Primeros casos de uso - Planificación
Elaboración	- Modelo de negocio - Requisitos funcionales y no funcionales - Características de los datos - Arquitectura software - Modelos del sistema
Construcción	- Producto software funcional - Manuales de usuario e instalación - Documentación
Transición	- Sistema probado y validado - Conclusiones y *Lessons Learned* - Trabajo futuro

Tabla 2: Proceso Unificado – outputs

Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
Trabajo Fin de Grado	**69,63 días**	**lun 21/03/16**	**mié 10/08/16**	
Fase 1: Inicio	**4,25 días**	**lun 21/03/16**	**mar 29/03/16**	
Elaboración Planificación	20 horas	lun 21/03/16	sáb 26/03/16	
Redacción de breve descripción	3 horas	sáb 26/03/16	sáb 26/03/16	3
Investigación de aplicaciones similares	8 horas	sáb 26/03/16	mar 29/03/16	4
Definir alcance	3 horas	mar 29/03/16	mar 29/03/16	5
Documentación	32 horas	lun 21/03/16	mar 29/03/16	

Figura 8: Fase de inicio

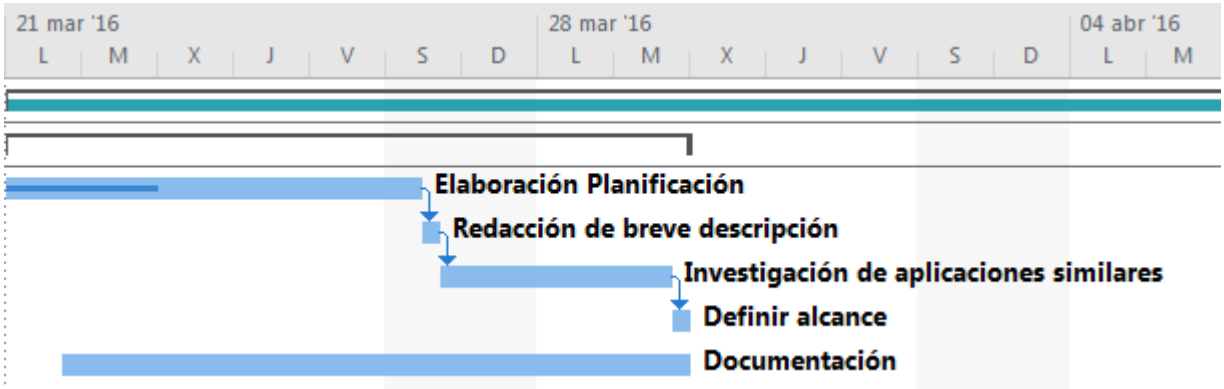


Figura 9: Fase de Inicio, diagrama de Gantt

3.1.2 Fase de elaboración

Esta fase englobará todo el proceso de análisis del proyecto. Esto se hará en una primera parte de identificación y definición de requisitos refinada: requisitos de usuarios, del sistema, funcionales y no funcionales. De esta subfase se obtendrá la lista de todos los requisitos del sistema, a partir de la cual se elaborarán los casos de uso que guiarán el desarrollo de todas las funciones y restricciones que debe ofrecer el sistema en su totalidad.

Con los requisitos y los casos de uso se elaborará la arquitectura más general del sistema. Los subsistemas y componentes que lo formarán y las interfaces entre los mismos. Una vez estén definidos los modelos arquitectónicos y el modelo de casos de uso, estos se retroalimentarán mutuamente en cada iteración del desarrollo dando lugar en cada una de las iteraciones a un prototipo funcional, que irá incrementándose periódicamente hasta que se dé con la solución definitiva.

En la Figura 10 se encuentra el desglose en subfases de esta fase de elaboración junto con los tiempos estimados y en la Figura 11 el diagrama de Gantt resultante.

Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
Fase 2: Elaboración	**11,63 días**	**mié 30/03/16**	**sáb 23/04/16**	
Ingeniería requisitos	**5,63 días**	**mié 30/03/16**	**sáb 09/04/16**	**2**
Documentación	5 días	mié 30/03/16	sáb 09/04/16	
Redacción del problema detallada	4 horas	mié 30/03/16	jue 31/03/16	
Especificación requisitos de usuario	**1,25 días**	**jue 31/03/16**	**sáb 02/04/16**	**11**
Identificación requisitos de usuario prelimina	3 horas	jue 31/03/16	jue 31/03/16	
Revisión de requisitos de usuario	1 hora	vie 01/04/16	vie 01/04/16	13
Documentación de requisitos de usuario	6 horas	vie 01/04/16	sáb 02/04/16	14
Especificación requisitos de sistema	**1,13 días**	**sáb 02/04/16**	**lun 04/04/16**	**12**
Identificación requisitos de sistema prelimina	3 horas	sáb 02/04/16	sáb 02/04/16	
Revisión de requisitos de sistema	2 horas	sáb 02/04/16	sáb 02/04/16	17
Documentación de requisitos de sistema	4 horas	sáb 02/04/16	lun 04/04/16	18
Elaboración de casos de uso	10 horas	lun 04/04/16	jue 07/04/16	16
Redacción documento requisitos	12 horas	jue 07/04/16	sáb 09/04/16	20
Documento de requisitos	0 días	sáb 09/04/16	sáb 09/04/16	21
Análisis	**6 días**	**sáb 09/04/16**	**sáb 23/04/16**	**9**
Documentación	4 días	sáb 09/04/16	lun 18/04/16	
Elaboración de modelos del sistema	**2,5 días**	**sáb 09/04/16**	**sáb 16/04/16**	
Elaboración de modelos estáticos	**1,25 días**	**sáb 09/04/16**	**mié 13/04/16**	
Modelos de Dominio	10 horas	sáb 09/04/16	mié 13/04/16	
Elaboración de modelos dinámicos	**1,25 días**	**mié 13/04/16**	**sáb 16/04/16**	
Modelos de flujo de datos	10 horas	mié 13/04/16	sáb 16/04/16	27
Especificación formal	**1,5 días**	**sáb 16/04/16**	**lun 18/04/16**	**25**
Especificación de subsistemas	6 horas	sáb 16/04/16	sáb 16/04/16	
Especificación de interfaces	6 horas	sáb 16/04/16	lun 18/04/16	31
Redacción documento de análisis	16 horas	lun 18/04/16	sáb 23/04/16	30
Documento de análisis	0 días	sáb 23/04/16	sáb 23/04/16	33

Figura 10: Fase de elaboración

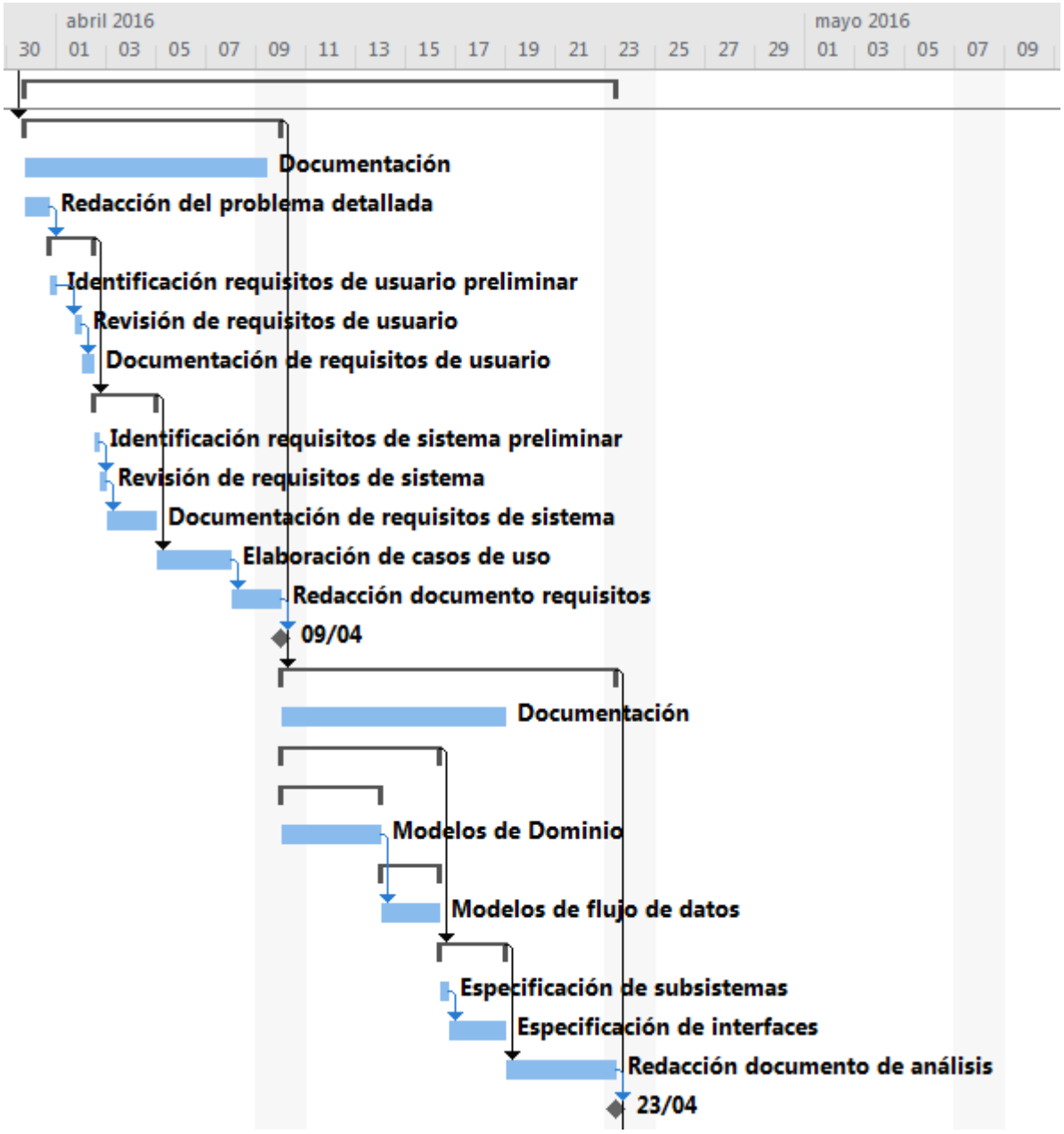


Figura 11: Fase de elaboración, diagrama de Gantt

3.1.3 Fase de construcción

En esta fase se elaborarán los modelos del sistema detallados junto con sus justificaciones. Estos modelos serán los modelos estáticos (modelo de dominio) y dinámicos (diagramas de secuencia). Así como los diseños de la interfaz de usuario y de la base de datos.

A partir de estos modelos se implementará el código fuente de la aplicación, dando como resultado el sistema software funcional acorde con los requisitos y los casos de uso. Al ser un proceso iterativo e incremental se irán desarrollando prototipos que irán mejorándose en calidad y completitud en cada una de las iteraciones.

En la Figura 12 se encuentran las subfases que componen esta fase de construcción junto con sus tiempos y en la Figura 13 el diagrama de Gantt resultante.

Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
Fase 3: Construcción	**19 días**	**sáb 23/04/16**	**mié 01/06/16**	**23**
Diseño	**7,25 días**	**sáb 23/04/16**	**sáb 07/05/16**	
Documentación	3,5 días	sáb 23/04/16	sáb 30/04/16	
Diseño arquitectónico	**4,38 días**	**sáb 23/04/16**	**sáb 30/04/16**	
Definición organización del sistema	**1,63 días**	**sáb 23/04/16**	**mar 26/04/16**	
Redacción de la estructura del sistema	5 horas	sáb 23/04/16	sáb 23/04/16	
Elaboración de modelos gráficos	8 horas	sáb 23/04/16	mar 26/04/16	40
Definición de descomposición modular	**0,5 días**	**mar 26/04/16**	**mié 27/04/16**	**39**
Definición del estilo de control	**0,5 días**	**mié 27/04/16**	**jue 28/04/16**	**42**
Diseño orientado a objetos	**0,75 días**	**jue 28/04/16**	**sáb 30/04/16**	**44**
Identificación de objetos	2 horas	jue 28/04/16	vie 29/04/16	
Elaboración de modelos	4 horas	vie 29/04/16	sáb 30/04/16	47
Diseño de la DB	**1 día**	**sáb 30/04/16**	**sáb 30/04/16**	**46**
Identificación de tablas relacionales	2 horas	sáb 30/04/16	sáb 30/04/16	
Identificación de relaciones	1 hora	sáb 30/04/16	sáb 30/04/16	50
Elaboración del modelo E-R	3 horas	sáb 30/04/16	sáb 30/04/16	51
Validación	2 horas	sáb 30/04/16	sáb 30/04/16	52
Diseño de la interfaz de usuario	**1,38 días**	**lun 02/05/16**	**jue 05/05/16**	**38**
Análisis de usuarios	3 horas	lun 02/05/16	lun 02/05/16	
Prototipado de la interfaz de usuario	8 horas	lun 02/05/16	jue 05/05/16	55
Evaluación de la interfaz de usuario	1 hora	lun 02/05/16	lun 02/05/16	
Redacción de Documento de Diseño	12 horas	jue 05/05/16	sáb 07/05/16	54
Documento de Diseño	0 días	sáb 07/05/16	sáb 07/05/16	58
Implementación	**19 días**	**sáb 23/04/16**	**mié 01/06/16**	
Aprendizaje Tecnologías	40 horas	sáb 23/04/16	mar 03/05/16	
Instalación y connfiguración de sistemas auxiliar	6 horas	mar 03/05/16	jue 05/05/16	61
Implementación DB	6 horas	jue 05/05/16	vie 06/05/16	62
Código fuente	100 horas	vie 06/05/16	mié 01/06/16	63

Figura 12: fase de construcción

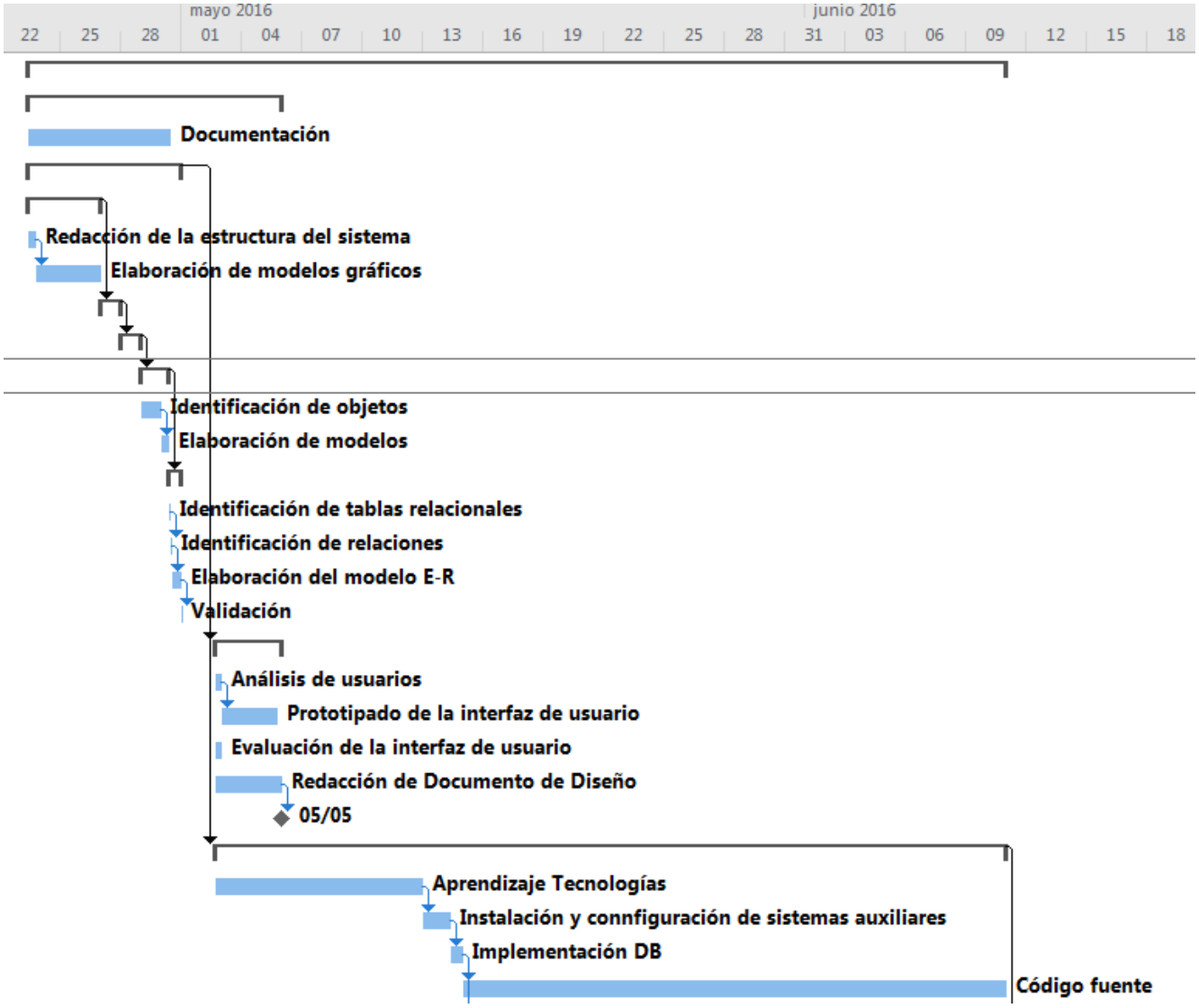


Figura 13: Fase de construcción, diagrama de Gantt

3.1.4 Fase de transición

De esta última fase, en la que se probará y validará el producto, se obtendrán una serie de lecciones aprendidas (Lessons Learned) junto con las conclusiones obtenidas, así como las líneas de trabajo futuro en la herramienta: posibles funcionalidades adicionales o mejoras. Ver Figura 14 y Figura 15.

Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
Fase 3: Construcción	**19 días**	**sáb 23/04/16**	**mié 01/06/16**	**23**
Fase 4: Transición	**20 días**	**vie 06/05/16**	**vie 17/06/16**	
Pruebas	100 horas	vie 06/05/16	mié 01/06/16	63
Elaboración de la memoria	60 horas	mié 01/06/16	vie 17/06/16	60
Memoria terminada	0 días	vie 17/06/16	vie 17/06/16	67

Figura 14: Fase de transición

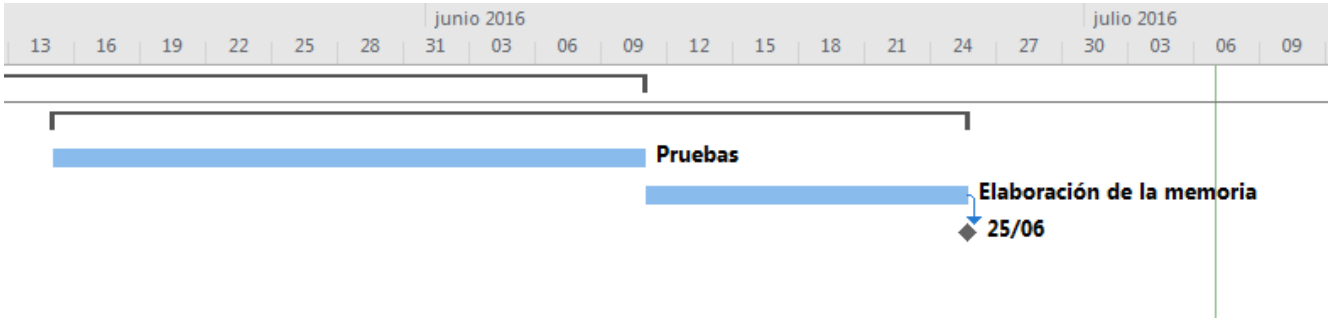


Figura 15: Fase de transición, diagrama de Gantt

3.2 COSTES

3.2.1 Herramientas Software utilizadas

En la medida de lo posible se emplearán ante todo programas con licencia gratuita, o bien, con licencia educativa proporcionada por la propia Universidad de Valladolid con el objetivo de minimizar el impacto económico del trabajo.

La relación de software comercial empleado para el desarrollo de este trabajo se presenta en la Tabla 3, donde se obviarán los productos con licenciamiento público o gratuito. A pesar de ser software comercial se han empleado versiones con licencia educativa, por lo que el coste total del software ha sido de 0€:

Software	**Coste**
Windows 7 Professional x64	Licencia educativa – 0€
Microsoft Office 2016	Licencia educativa – 0€
Microsoft Project 2016	Licencia educativa – 0€
Astah Professional	Licencia educativa – 0€
Total	**0 €**

Tabla 3: Relación de costes de software comercial

3.2.2 Herramientas Hardware utilizadas

Para generar la aplicación web, así como la documentación se ha empleado el sistema que se detalla en la Tabla 4. Se ha utilizado también un dispositivo móvil smartphone como hardware de pruebas. Sus características se reflejan en la Tabla 5.

HP Pavilion g6 1203ss (15 pulgadas, principios 2012)	
Procesador	Intel Core i5-2430M @ 2.40GHz (2 núcleos)
Memoria Cache	3 MB
Memoria	8 GB
Disco duro	500 GB
Tarjeta gráfica	Intel HD Graphics 3000
Pantalla	1366 x 768

Tabla 4: Características HP Pavilion g6

Samsung Galaxy S4 Mini gt-i9195	
Procesador	Dual-core 1.7 GHz Krait 300
Memoria	1.5 GB
Disco duro	8 GB
Pantalla	540 x 960

Tabla 5: Características Samsung Galaxy S4 Mini

Así mismo los costes de los recursos hardware empleados se presentan en la Tabla 6. Se considerará una amortización de los equipos empleados en 2 años. Por lo que la fórmula empleada para el cálculo del coste ha sido la siguiente, donde p es el precio total del elemento y n el número de meses que ha sido utilizado durante el trabajo:

$$Coste = \frac{p}{24} \times n$$

Hardware	Coste total	Número de meses	Coste amortizado
Portátil HP Pavilion g6 1202-ss	519.95 €	4	**86.66 €**
Samsung Galaxy S4 Mini gt-i9195	199.00 €	4	**33.17 €**

Tabla 6: Relación de costes hardware

3.2.3 Recursos Humanos

Se hará además una estimación de costes en base al trabajo realizado por la persona encargada del proceso de planificación y desarrollo de este proyecto. Se hará una separación de roles dependiendo de la función específica de dicha persona y cuántas horas se han necesitado dentro de cada rol de forma resumida en la Tabla 7. Todas las funciones han sido desempeñadas por la misma persona, pero en esta tabla se trata reflejar de una forma más o menos cercana a la realidad cómo se estimaría un proyecto software de estas características. El número de horas considerado en la tabla se ha calculado a partir de la planificación anteriormente definida.

Los precios se han calculado tomando como referencia las retribuciones que perciben los empleados públicos de España según el Boletín Oficial del Estado del año 2016 [23].

Rol / Función	Grupo	Nº de horas	Coste / hora	Coste total
Analista - Diseñador	A1	160	20.75 €	**3320.53 €**
Técnico de sistemas	B	18	14.3 €	**257.45 €**
Programador senior	A2	152	16.3 €	**2482.68 €**
Programador junior (pruebas)	C1	100	12.5 €	**1254.43 €**

Tabla 7: Relación de costes en recursos humanos

3.3 RIESGOS

El plan de gestión de riesgos está concebido para guiar a identificar, analizar y gestionar los riesgos que se puedan producir durante la etapa de desarrollo del sistema de gestión de incidencias. Se empleará una estrategia sistémica, de tal forma que se identifiquen los riesgos potenciales teniendo en cuenta su probabilidad de ocurrencia, así como su posible impacto.

3.3.1 Definición de los riesgos

En la tabla se identifican una serie de riesgos que podrán aparecer a lo largo del proceso de desarrollo. Mediante esta identificación inicial se hará un plan de gestión de riesgos para minimizar el impacto que estos puedan tener sobre el proyecto.

La definición de cada uno de los riesgos irá dada por:

- Nombre: Nombre identificativo para el riesgo.
- Descripción: Descripción breve del riesgo a tratar.
- Probabilidad: Probabilidad de ocurrencia del riesgo. Valores: baja, media, alta, muy alta.
- Impacto: Impacto que puede ocasionarse si el riesgo ocurre. Valores: bajo, medio, alto, muy alto.
- Riesgo: resultado de combinar la probabilidad con el impacto (ver matriz de probabilidad – impacto, tabla 9).
- Estrategia de prevención: Acciones para evitar que el riesgo aparezca, y en caso de aparecer, minimizar su impacto. Serán accionas aplicadas antes de que el riesgo pueda aparecer.
- Plan de contingencia: Acciones realizadas una vez el riesgo ha aparecido con el objetivo de corregir el impacto ocasionado.

3.3.2 Riesgos

# Riesgo	RI-01
Nombre	Planificación optimista
Descripción	Una planificación inicial que no es acorde a la realidad final del desarrollo
Probabilidad	Alta
Impacto	Medio
Riesgo	Alto
Estrategia de prevención	Realización de una planificación lo más detallada posible dentro del tiempo establecido y con revisiones periódicas
Plan de contingencia	Reorganización del trabajo

Tabla 8: Riesgo: Planificación optimista

# Riesgo	RI-02
Nombre	Pérdida de datos
Descripción	Debido a un evento accidental se produce la pérdida de parte del proyecto
Probabilidad	Baja
Impacto	Alto
Riesgo	Medio
Estrategia de prevención	Mantener copias de seguridad actualizadas de los datos en otros soportes: discos externos y cloud (OneDrive)
Plan de contingencia	Recuperar la copia de seguridad y, en caso necesario, realizar de nuevo el trabajo perdido

Tabla 9: Riesgo: Pérdida de datos

# Riesgo	RI-03
Nombre	Avería de equipos
Descripción	El equipo con el que se desarrolla el producto se avería (la pérdida de datos está contemplada en el riesgo RI-01)
Probabilidad	Baja
Impacto	Alto
Riesgo	Medio
Estrategia de prevención	Documentar el proceso de instalación y configuración del software que conforma la infraestructura sobre la que se sustenta el despliegue del producto.
Plan de contingencia	Cambio del equipo y reinstalación y configuración de la infraestructura software

Tabla 10: Riesgo: Avería de equipos

# Riesgo	RI-04
Nombre	Falta de disponibilidad del personal
Descripción	Falta de disponibilidad no prevista en la planificación por parte del autor o el tutor del Trabajo de Fin de Grado
Probabilidad	Alta
Impacto	Medio
Riesgo	Alto
Estrategia de prevención	Realización de una planificación lo más realista posible dentro del tiempo establecido
Plan de contingencia	Reorganización del trabajo

Tabla 11: Riesgo: Falta de disponibilidad del personal

# Riesgo	RI-05
Nombre	Modificación de requisitos
Descripción	Modificación en la definición de los requisitos en fases avanzadas del proyecto
Probabilidad	Media
Impacto	Alto
Riesgo	Alto
Estrategia de prevención	Utilización de patrones de diseño que minimicen el impacto a la hora de modificar el modelo de diseño. Revisión de los requisitos en cada iteración.
Plan de contingencia	Replanificar y realizar el trabajo que implique la modificación de requisitos

Tabla 12: Riesgo: Modifiicación de requisitos

# Riesgo	RI-06
Nombre	Presupuesto insuficiente
Descripción	No se dispone del presupuesto económico necesario para el desarrollo del proyecto
Probabilidad	Baja
Impacto	Medio
Riesgo	Bajo
Estrategia de prevención	Utilización de herramientas o productos con licenciamiento público o educativo en todos los casos en los que sea posible
Plan de contingencia	Buscar una vía de ingresos para suplir la carencia económica

Tabla 13: Riesgo: Presupuesto insuficiente

3.3.3 Matriz de Probabilidad - Impacto

En la Tabla 14 se clasificarán los riesgos definidos en función de su probabilidad de ocurrencia y el impacto que puede tener para el proyecto en general. De forma gráfica se visualizan los riesgos cuya prevención se debe priorizar dadas sus características. Así, en color rojo aparecerán los riesgos considerados críticos, en naranja los riesgos altos y en verde, los riesgos asumibles.

		Impacto			
		Bajo	Medio	Alto	Muy Alto
Probabilidad	Baja		RI-06	RI-02 RI-03	
	Media			RI-05	
	Alta		RI-01 RI-04		
	Muy Alta				

Tabla 14: Matriz Probabilidad – Impacto

3.3.4 Resumen

En este bloque se ha presentado la planificación del trabajo, junto con el análisis y la gestión de los riesgos identificados. En el Bloque VIII se hará una valoración final del cumplimiento de esta estimación de trabajo, así como de los costes asociados a ella.

4 BLOQUE IV:
ANÁLISIS DE REQUISITOS

En esta sección se recogerá toda la información obtenida en esta fase de ingeniería de requisitos, esto es, la especificación formal de requisitos y documentación correspondiente al análisis de la aplicación. De aquí se extraerán los subsistemas que conformarán el sistema al completo, así como su diseño, las restricciones que esté presentará, y las decisiones de diseño (y sus justificaciones) que se tomarán teniendo en cuenta la naturaleza del sistema.

Se comenzará por definir una serie de métricas que den un valor medible de alguna forma a ciertos adjetivos o consideraciones que se puedan emplear a lo largo del proceso de análisis.

4.1 MÉTRICAS

Para evitar vaguedades a la hora de dar juicios de valor a una serie de características funcionales de la aplicación, como por ejemplo decir que la página es *rápida* sin especificar qué se considera exactamente *rapidez* en este contexto, se elaborará una tabla que asociará cada característica con las medidas que se consideran buenas, aceptables o malas. La existencia de métricas ayudará además a elaborar un plan de calidad más preciso, basado en valores medibles para así poder facilitar un proceso de mejora continuada.

4.1.1 Rapidez:

Como rapidez se entiende el tiempo de uso en general al utilizar la aplicación al llevar a cabo una tarea dentro de la misma. En esta característica influyen varios factores: los tiempos de carga, el número de recargas / actualizaciones de la página, el número de clicks al ejecutar la tarea y el tiempo de reflexión por parte del usuario, así como factores como la congestión de la red y la latencia de los dispositivos de red. Se considerará un entorno ideal a la hora de hacer las pruebas, esto es, alojando en local la aplicación. Por tanto, se detallan a continuación cada una de las partes que conforman la rapidez.

4.1.1.1 Tiempo de carga

El tiempo de carga de la página corresponde al tiempo que transcurre desde que se hace una petición HTTP al servidor web, hasta que la respuesta es devuelta y presentada al usuario. Esta latencia puede variar dependiendo del volumen de información que se mueva en el proceso. Una carga de una página de presentación de cantidades altas de datos, como puede ser una tabla con todos los usuarios registrado, puede tardar mucho más que la presentación de un formulario sencillo.

Esta métrica se medirá en segundos y se utilizará para medirla el complemento de Mozilla Firefox Firebug. A continuación, los valores considerados buenos, aceptables y malos. Estos valores son tomados de la referencia bibliográfica *Usability Engineering* de Jakob Nielsen [23]. En ella se indica que un tiempo de respuesta de 0.1 s es el límite en el que un usuario aprecia que la tarea ha sido inmediata, este resultado se catalogará como bueno, hasta un tiempo de 1 s se considera que el usuario permanece pensando o concentrado en la misma tarea. Por último, un tiempo de 10 s es el límite que se estima que el usuario permanezca atento a la aplicación, pasado este tiempo, comenzará a realizar otras tareas mientras se espera al resultado, lo que romperá con la concentración del usuario en la tarea que está llevando a cabo.

Bueno (s)	Aceptable (s)	Malo (s)	Muy Malo (s)
0 – 0.1	0.1 - 1	1 - 10	Más de 10

Tabla 15: Tiempo de carga

4.1.1.2 Número de clics a la hora de ejecutar una tarea

Así mismo el número de clics puede ser una métrica útil al medir la velocidad de la aplicación basándose en la interacción con el usuario. Como clic se entiende el seleccionar un enlace, se obviarán los clics en los campos de los formularios ya que son clics que son inevitables y además se puede navegar a través de los campos mediante el tabulador (también se pueden seleccionar los enlaces con el tabulador, pero es algo mucho más tedioso ya que hay que navegar a través de todos los enlaces hasta dar con el adecuado, y el tiempo variará en función de la organización de los mismos). Se hará uso de una regla que, aunque no es oficial, es bastante

interesante y conocida por la comunidad; se trata de la regla de los 3 clics [24], consistente en que el usuario debería ser capaz de acceder a toda la información de la página con tan solo 3 clics a partir de la página inicial.

Bueno (s)	Aceptable (s)	Malo (s)
1-3	3-5	Más de 5

Tabla 16: Clicks por tarea

4.1.2 Robustez

Se considerará un sistema robusto aquel que es resistente a errores no controlados. La métrica utilizada será el número de errores y warnings que aparecen por pantalla o consola a lo largo de una sesión de un usuario cualquiera, utilizando el sistema de una forma aleatoria (realizando un caso de uso cualquiera), pero tratando de abarcar todas las funcionalidades posibles. Al final del desarrollo este valor debe ser minimizado al máximo posible. A continuación, la tabla correspondiente a esta métrica:

Bueno (s)	Aceptable (s)	Malo (s)
0	1-2	Más de 2

Tabla 17: Errores por sesión

4.2 Datos y Entidades

Se definirán aquí con qué datos y entidades abstractas trabajará la aplicación. Se dará forma así el contexto sobre el cual se sustentará todo el sistema. Se empezará por los usuarios.

4.2.1 Características del usuario

UVaHelpDesk será capaz de coordinar las peticiones de servicio de los usuarios con el trabajo del equipo de IT encargado de resolverlos, así como a los Managers realizar un seguimiento de todos los tickets que haya creados.

De esta forma se pueden diferenciar 3 tipos de usuario: el usuario final (cliente a partir de ahora), el técnico y el mánager del equipo de técnicos.

- **Cliente**: Se trata del usuario que necesita algún tipo de servicio técnico especializado en una determinada área. Sus funciones se limitarán, pues, a crear tickets y comprobar su estado. Este usuario podrá hacer login en la aplicación, crear tickets, realizar un seguimiento exclusivamente de los tickets en los que se vea afectado, cerrar tickets (darlos por resueltos) o reabrir un ticket (si se ha dado por cerrado y el problema persiste) y cambiar su contraseña.
- **Técnico**: Se trata del usuario encargado de dar solución a los diferentes tickets que se crean en el sistema. Este usuario puede hacer login en la aplicación, modificar el estado de un ticket (esto incluye darlo por resuelto), realizar búsquedas de los tickets en la base de datos. Para mantener una organización más controlable de los técnicos, se agruparán estos en Grupos que la aplicación permitirá crear de forma flexible y cambiar su contraseña.
- **Mánager**: Este usuario puede hacer login en la aplicación, modificar el estado de un ticket (esto incluye darlo por resuelto), asignar un técnico, reabrir un ticket, realizar búsquedas de los tickets en la base de datos y cambiar su contraseña.

Lo lógico es que cada usuario final tenga la posibilidad de tener un seguimiento de su propio ticket, pero no del resto de usuarios dado que no es información que sea relevante para alguien que esté realizando una petición de servicio informático. Por otro lado, esta información sí puede ser de utilidad para un técnico que quiera buscar casos similares para facilitar así la resolución, o para un mánager que necesite hacer un seguimiento de su equipo de trabajo o de las secciones que tiene bajo su control.

	Cliente	Técnico	Mánager
Login	Sí	Sí	Sí
Abrir ticket	Sólo los propios	Sólo los propios	Sólo los propios
Búsquedas tickets	Sólo los propios	Sí	Sí
Asignar técnico	No	No	Sí
Crear usuarios	No	No	Sí

Tabla 18: Funciones usuarios

4.2.2 Características del Ticket

El elemento fundamental de las comunicaciones dentro de la aplicación es el concepto de ticket. Un ticket será una instancia de petición de servicio desde un cliente hasta un técnico. Para cada ticket se tendrá en cuenta la información:

- **Fecha y hora de apertura**. Con la finalidad de realizar un seguimiento del cumplimiento de la SLA, así como dar información útil acerca del ticket en sí. De esta forma se podrán realizar búsquedas más afinadas
- **Creador**. Se relacionará el ticket con su usuario creador.
- **Técnico asignado**. De la misma forma se relacionará el ticket con el técnico que tiene asignado actualmente, o el técnico que lo resolvió en caso de estar cerrado el ticket.
- **Título**. Una breve descripción del problema a modo de resumen rápido a la hora de presentar la información en tablas, etc…
- **Descripción**. Descripción detallada del problema que le ha surgido al usuario que ha creado el ticket.
- **Solución**. Solución aportada por el técnico, con la finalidad de organizar cada ticket con su solución. Esto permitirá al resto de técnicos tener una guía de trabajo basada en tickets similares realizando búsquedas en la base de datos a modo de repositorio de tickets con sus soluciones.
- **Número identificativo**. Número único e identificativo para cada ticket.
- **Prioridad**. Prioridad con la que se considera que el ticket debe estar resuelto. Esto influirá en el cálculo de la SLA.
- **Tiempo transcurrido**. Tiempo transcurrido desde la creación del ticket y el momento actual. Es un dato interesante para ver qué tickets llevan abiertos más tiempo, etc…
- **Centro afectado**: El centro que se ha visto afectado por la incidencia en cuestión. Estos centros agruparán en secciones para mantener una mayor coherencia de los datos cuando la aplicación alcance un número importante de datos almacenados, de esta forma será más sencillo realizar búsquedas de centros si se conoce la sección a la que pertenece, o sacar estadísticas más detalladas para el mánager.

El Ticket deberá pasar por un ciclo de vida basado en estados, desde su creación, hasta su cierre definitivo. En la figura 19 se muestra de forma resumida este ciclo de vida de un ticket:

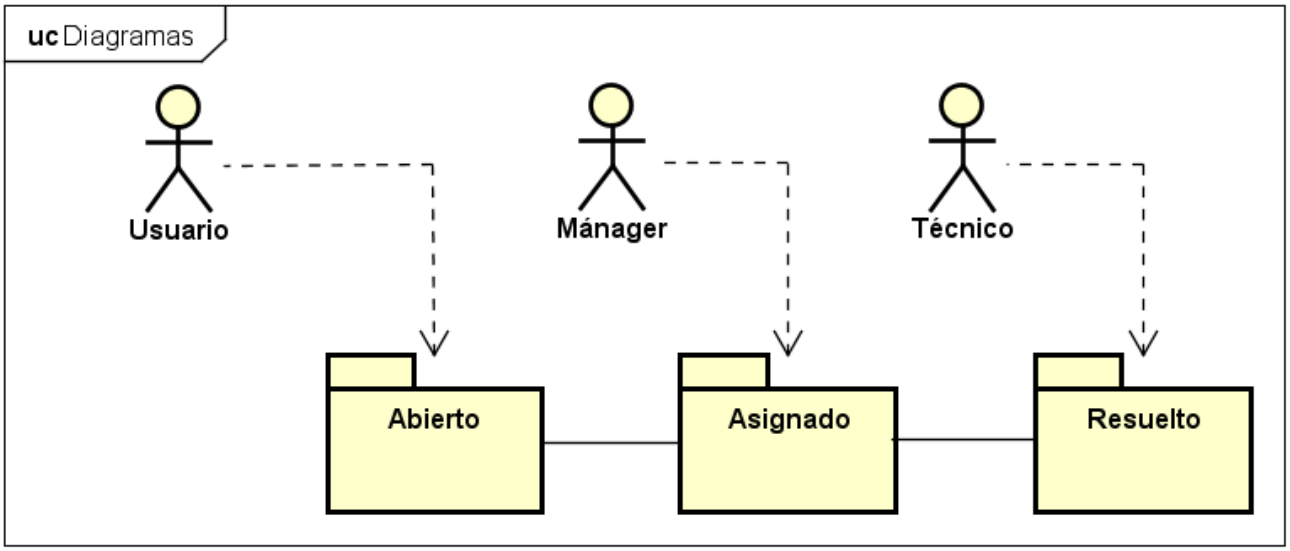


Figura 16: Ciclo de vida del Ticket

Los estados por los que pasará el Ticket en orden cronológico serán:

- **Abierto**: Ticket recién creado por un usuario cualquiera. Cualquier usuario puede crear tickets en su nombre, ya sea Cliente, Técnico o Mánager. Este Ticket aún no estará asignado; esa tarea quedará relegada al mánager que controla la sección a la cual el Ticket afecta, recordemos que una sección es una agrupación de Centros.
- **Asignado**: Una vez el mánager ha asignado un técnico al Ticket, este pasará a estado *Asignado*. Por lo tanto, el mánager debe tener la opción de ver los tickets abiertos de su sección para posteriormente asignárselos al técnico que considere adecuado. Así mismo, el técnico que ha recibido un ticket para resolver debe ser debidamente notificado de dicha asignación. Una vez lo resuelva, cambiará el estado del mismo a "*Resuelto*" indicando la solución empleada.
- **Resuelto**: Una vez el ticket ha sido resuelto, el cliente que lo ha abierto tendrá la oportunidad de volver a abrirlo si considera que el problema persiste, volviendo al estado "Abierto", o bien darlo por cerrado definitivamente si considera que su problema ha sido solucionado. Si al cabo de 7 días no se ha realizado ningún cambio de estado de los mencionados, el ticket pasará por sí mismo al estado de "*Cerrado*".
- **Cerrado**: En este momento el ticket se ha dado por cerrado definitivamente y no aceptará más cambios de estado ni modificaciones. Quedará como testigo de una petición de servicio terminada satisfactoriamente y a modo de repositorio para futuras consultas por parte de los técnicos que busquen casos similares, mánagers para ejercer tareas de control o administración de sus recursos, etcétera.

Cada una de las modificaciones que se hagan sobre un Ticket quedarán registradas en el sistema, para facilitar el seguimiento del ciclo de vida del mismo por parte de los mánagers. Como modificaciones se entenderán:

- Aperturas de Tickets por parte de los usuarios.
- Cambios de estado.
- Reasignaciones de técnicos.

Cada vez que se modifique un Ticket, se dará la oportunidad de agregar un comentario con el objetivo de ofrecer Feedback. Por ejemplo, en una reasignación de técnico puede ser interesante incluir en dicha modificación la justificación de la misma.

4.2.3 Características de los Centros y las Secciones

Como se ha comentado, los tickets estarán vinculados con los centros que se han visto afectados por algún tipo de incidencia o petición de servicio técnico. Estos centros estarán categorizados por Secciones, con el fin de

facilitar las tareas cuando el volumen de datos sea muy elevado. También se contemplará la posibilidad de incluir gráficas o estadísticas para las labores de control o auditoría que tendrán los mánagers.

4.3 REQUISITOS

Esta parte del documento será el pilar más sólido del proceso de desarrollo formalmente hablando. Ya se ha elaborado una versión inicial de la especificación del producto, pero todas las decisiones de diseño se extraerán en primera instancia de este documento, más descriptivo, formal y concreto. De esta forma el público al que está dirigido es el mismo desarrollador, como herramienta de apoyo; y al posible tribunal que evalúe este trabajo.

UVaHelpDesk es un producto que para alguna de sus funciones dependerá de otras herramientas ya implementadas, siempre de licencia gratuita, tales como sistemas gestores de bases de datos, librerías de lenguajes de programación (APIs), servidores web o alojamiento remoto, pero estas decisiones se tomarán a posteriori, de momento el documento girará en torno a lo que se debe lograr, dejando el cómo para la fase de diseño.

El trabajo comprenderá el análisis, diseño e implementación del código fuente de parte del servidor que, uniendo estos subsistemas, permita dar solución al problema que se plantea.

4.3.1 Requisitos Funcionales

Dado que la funcionalidad de la aplicación depende en gran medida del tipo de usuario que esté utilizándola, se desglosará la funcionalidad en los diferentes tipos de usuario. Se dará una referencia a cada uno de los requisitos para futuras referencias en el documento.

Cliente:

- **RF-01**: Se permitirá hacer login al usuario por medio de su nombre de usuario y su contraseña. Se informará en caso de fallo y se entrará en la vista principal en caso correcto.
- **RF-02**: Deberá tener la posibilidad de abrir un ticket aportando los datos: título de la petición, descripción, prioridad, sección y centro afectados. El sistema generará un número de ticket único e identificativo.
- **RF-03**: Deberá poder visualizar los tickets que hay abiertos, o han estado abiertos, a su nombre.
- **RF-04**: Podrá ver los detalles de un ticket.
- **RF-05**: Podrá cambiar su contraseña.
- **RF-06**: Podrá cerrar su sesión.
- **RF-07**: Podrá consultar su perfil de usuario.
- **RF-08**: Podrá cambiar su imagen de perfil.

Técnico:

- **RF-09**: Se permitirá hacer login al usuario por medio de su nombre de usuario y su contraseña. Se informará en caso de fallo y se entrará en la vista principal en caso correcto.
- **RF-10**: Deberá tener la posibilidad de abrir un ticket aportando los datos: título de la petición, descripción, prioridad, usuario afectado. El sistema generará un número de ticket único e identificativo.
- **RF-11**: Podrá ver los detalles de un ticket, su historial de modificaciones, y el centro afectado.
- **RF-12**: Deberá poder visualizar o realizar búsquedas en los tickets que hay abiertos, o han estado abiertos de todo el sistema.
- **RF-13**: Podrá cambiar su contraseña.
- **RF-14**: Podrá cerrar la sesión.
- **RF-15**: Podrá visualizar los tickets que tiene asignados en un determinado momento.
- **RF-16**: Podrá cambiar el estado de los tickets asignados y modificar la solución.
- **RF-17**: Podrá consultar su perfil de usuario.
- **RF-18**: Podrá cambiar su imagen de perfil.

Mánager:

- **RF-19**: Se permitirá hacer login al usuario por medio de su nombre de usuario y su contraseña. Se informará en caso de fallo y se entrará en la vista principal en caso correcto.
- **RF-20**: Deberá tener la posibilidad de abrir un ticket aportando los datos: título de la petición, descripción, prioridad, usuario afectado, técnico asignado. El sistema generará un número de ticket único e identificativo.
- **RF-21**: Deberá poder visualizar o realizar búsquedas en los tickets que hay abiertos, o han estado abiertos de todo el sistema.
- **RF-22**: Podrá ver los tickets que están sin asignar de secciones bajo su control.
- **RF-23**: Podrá ver los detalles de un ticket, su historial de modificaciones, el técnico que tiene asignado y el centro afectado.
- **RF-24**: Podrá asignar y modificar el técnico asignado en cualquier ticket.
- **RF-25**: Podrá modificar el estado de cualquier ticket.
- **RF-26**: Podrá cambiar su contraseña.
- **RF-27**: Podrá cerrar la sesión.
- **RF-28**: Deberá tener la posibilidad de añadir usuarios al sistema indicando si es técnico, cliente o mánager y los datos: email, teléfono, nombre y apellidos, centro en el caso de ser un cliente o mánager responsable y grupo en caso de ser un técnico.
- **RF-29**: Podrá consultar su perfil de usuario.
- **RF-30**: Podrá obtener un listado con los usuarios registrados.
- **RF-31**: Podrá obtener un listado con las secciones bajo su control.
- **RF-32**: Podrá crear centros aportando los datos: sección a la que pertenece, nombre, teléfono y dirección.
- **RF-33**: Podrá cambiar su imagen de perfil.
- **RF-34**: Podrá ver una serie de gráficas estadísticas con datos relevantes para la gestión.

Sistema:

- **RF-35**: Internamente se contará el tiempo transcurrido para cada ticket para cumplir con la SLA establecida. Se notificará visualmente al usuario el estado de cumplimentación con la SLA en la que se encuentra el ticket (Con diferentes fuentes o colores en el texto).
- **RF-36**: Se dará por cerrado automáticamente un Ticket que ha pasado 7 días en estado "*Resuelto*".
- **RF-37**: Limitará el tiempo límite de una sesión de usuario a 600 segundos por defecto.
- **RF-38**: La conexión debe ser segura, debe ser imposible obtener datos de las transacciones por medio de un rastreador de red o *sniffer*.

4.3.2 Requisitos No Funcionales

4.3.2.1 Seguridad

Debe ser una aplicación segura tanto en las transacciones de datos como en el almacenamiento de la información más sensible como pueden ser las contraseñas.

4.3.2.2 Usabilidad

La aplicación debe ser intuitiva y fácil de utilizar, aprender y memorizar sin un esfuerzo intenso por parte del usuario, tenga o no conocimientos de informática.

4.3.2.3 Robustez

Debe ser un sistema resistente a errores, que no de condiciones de error incontroladas en la medida de lo posible, para ello se hará hincapié en la validación de los formularios que requieran una entrada de datos controlada (números de teléfono, emails, cifras, fechas…).

4.3.2.4 Escalabilidad

Se hará uso de patrones de diseño reconocidos por la comunidad y bien documentados para facilitar la integración con otros sistemas y la escalabilidad futura del producto.

4.3.2.5 Interfaz de usuario

La interfaz de usuario será intuitiva, esto es, que el usuario no necesite acudir a un manual de usuario de forma frecuente, de rápido aprendizaje haciendo uso de texto acompañados de gráficos para facilitar la memoria fotográfica por parte del usuario a la hora de realizar las tareas y que termine siendo un proceso mecánico de forma más rápida.

No se sobrecargará la página con datos innecesarios, en caso de necesitar explicar algo que sea a través de mensajes emergentes al pasar el cursor evitando llenar la página de datos a priori innecesarios para el funcionamiento de la herramienta.

4.4 CASOS DE USO

De este análisis de requisitos previo se obtendrán una serie de Casos de Uso, más cercanos al diseño final de la aplicación, que encapsularán las diferentes funcionalidades que ejercerá el sistema.

En el diagrama de la Figura 17 se pueden ver de forma gráfica la serie de casos de uso identificados para la aplicación, a continuación, se muestra la definición formal para cada uno de ellos.

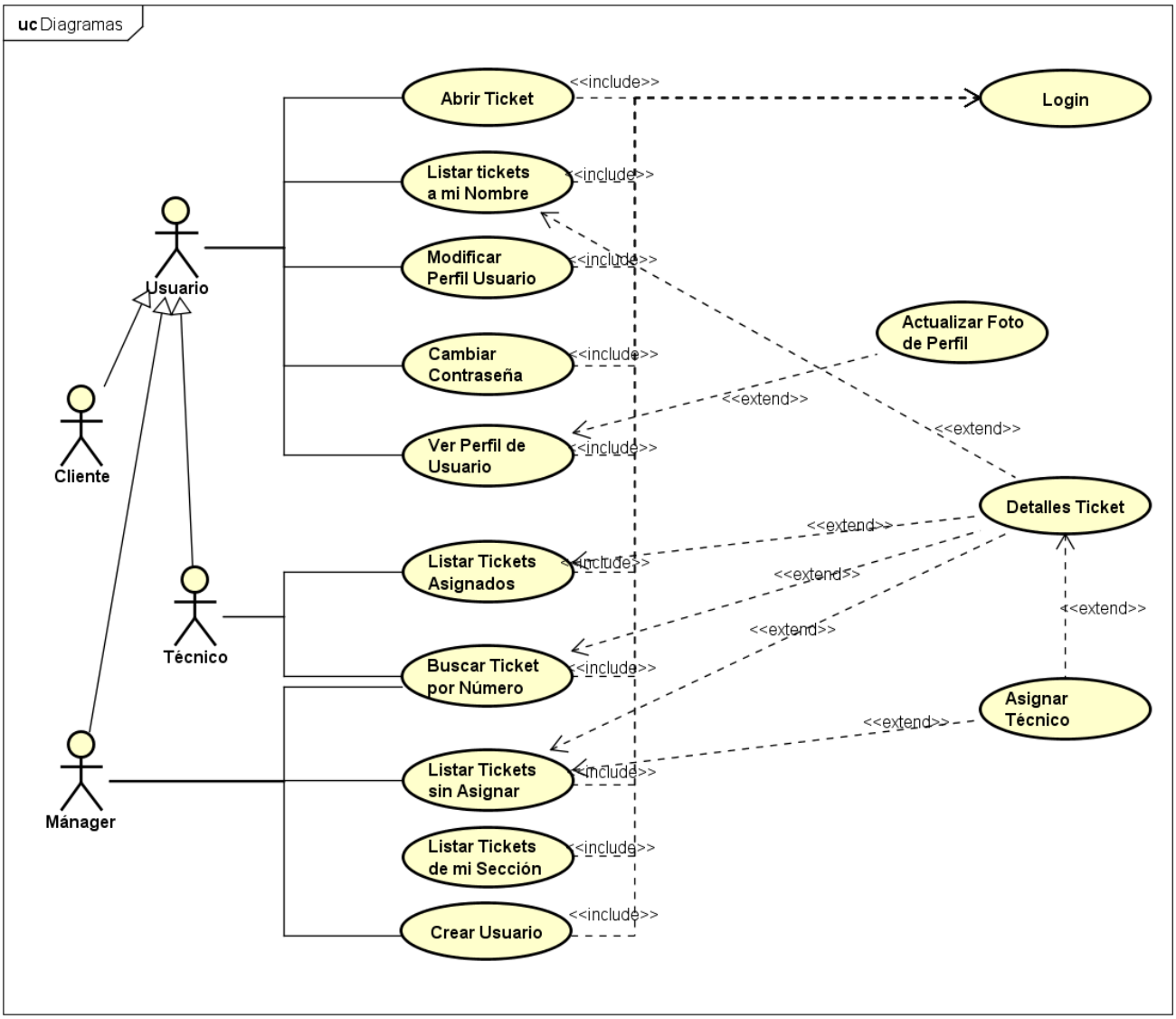


Figura 17: Diagrama de casos de uso

4.4.1 Login

Caso de Uso UC1: Login

Actor Principal: Usuario

Personal involucrado y lista de intereses:

- Cliente: Quiere entrar en la aplicación y que esta le muestre la vista adecuada a su condición de Cliente.
- Técnico: Quiere entrar en la aplicación y que esta le muestre la vista adecuada a su condición de Técnico.
- Mánager: Quiere entrar en la aplicación y que esta le muestre la vista adecuada a su condición de Mánager.

Precondiciones: ninguna

Postcondiciones: El Usuario entra en la aplicación. La aplicación muestra la vista adecuada dependiendo si es Cliente, Técnico o Mánager.

Flujo Básico:

1. El Usuario accede a través de la URL a la página inicial de la aplicación.
2. El Usuario introduce su nombre de usuario y su contraseña.
3. El Sistema valida si los datos introducidos concuerdan con los almacenados.
4. El Sistema presenta la interfaz de usuario correspondiente, en una página inicial de la aplicación.

Flujos Alternativos:

3a. Los datos no concuerdan con los almacenados:

1. El sistema notificará al usuario del error y vuelve a la página de login.

3b. El sistema encuentra algún fallo al comunicarse con el servicio externo que almacena los datos:

1. El sistema notificará del fallo concreto al usuario y volverá a la pantalla inicial.

4a. El Usuario es de tipo Cliente:

1. El Sistema presentará un formulario para crear un Ticket.

Caso de Uso 1: Login

4.4.2 Abrir Ticket

Caso de Uso UC2: Abrir Ticket

Actor Principal: Usuario

Personal involucrado y lista de intereses:

- Cliente: Quiere la posibilidad de abrir tickets en el sistema para notificar peticiones de servicio a los Técnicos.
- Técnico: Quiere que los Clientes abran tickets para desarrollar su trabajo, así como la creación de tickets a nombre de otros Clientes.
- Mánager: Quiere gestionar la asignación de tickets entre los técnicos, así como la creación de tickets a nombre de otros Clientes.

Precondiciones: El usuario está autenticado en el sistema. **Postcondiciones**: El sistema ha almacenado un ticket con los datos introducidos por el usuario. **Flujo Básico**:

1. El usuario accede a la opción de abrir un ticket.
2. El sistema presenta un formulario con datos para rellenar.
3. El usuario introduce los datos en el formulario y selecciona la opción de enviar.
4. El sistema comprueba que los datos son válidos.
5. El sistema almacena persistentemente el ticket, asignando un número único e identificativo.
6. El sistema notifica al usuario de la correcta creación del ticket.

Flujos Alternativos:

2a. El usuario es de tipo Cliente:

1. El sistema presentará un formulario con los campos editables: Título, Descripción, Prioridad.

2b. El usuario es de tipo Técnico:

1. El sistema presentará un formulario con los campos editables: Título, Descripción, Prioridad, Usuario Afectado.

2c. El usuario es de tipo Mánager:

1. El sistema presentará un formulario con los campos editables: Título, Descripción, Prioridad, Usuario Afectado, Técnico Asignado.

4a. Los datos son inválidos:

1. El sistema notificará del fallo y resaltará los campos inválidos en el formulario.

Caso de Uso 2: Abrir Ticket

4.4.3 Listar tickets a mi nombre

Caso de Uso UC3: Listar Tickets a mi Nombre

Actor Principal: Usuario

Personal involucrado y lista de intereses:

- Usuario: Quiere obtener un listado de los tickets creados a su nombre.

Precondiciones: El Usuario está autenticado en el sistema. **Postcondiciones**: El Sistema ha mostrado al Usuario una lista con todos los tickets a su nombre.

Flujo Básico:

1. El Usuario accede a la opción de listar los tickets a su nombre.
2. El Sistema presenta al Usuario una lista de todos los tickets almacenados a su nombre.

Flujos Alternativos:

2a. El Usuario no tiene ningún ticket a su nombre.

1. El Sistema presentará un mensaje de notificación en lugar de la lista.

Caso de Uso 3: Listar tickets a mi nombre

4.4.4 Listar tickets asignados

Caso de Uso UC4: Listar Tickets Asignados

Actor Principal: Técnico

Personal involucrado y lista de intereses:

- Técnico: Quiere obtener un listado de los tickets que tiene asignados y sin resolver en ese instante.

Precondiciones: El Usuario está autenticado en el sistema. **Postcondiciones**: El Sistema ha mostrado al Técnico una lista con todos los tickets asignados y sin resolver.

Flujo Básico:

1. El Técnico accede a la opción de listar los tickets asignados.
2. El Sistema presenta al Técnico una lista de todos los tickets asignados y sin resolver.

Flujos Alternativos:

2a. El Usuario no tiene ningún ticket asignado a su nombre.

1. El Sistema presentará un mensaje de notificación en lugar de la lista.

Caso de Uso 4: Listar Tickets Asignados

4.4.5 Listar Tickets de mi Sección

Caso de Uso UC5: Listar Tickets de mi Sección

Actor Principal: Mánager

Personal involucrado y lista de intereses:

- Mánager: Quiere obtener un listado de los tickets de su sección y sin resolver en ese instante, para poder asignárselos a un técnico.

Precondiciones: El Usuario está autenticado en el sistema. **Postcondiciones**: El Sistema ha mostrado al Usuario una lista con los tickets de su sección, y la posibilidad de asignárselos a los diferentes técnicos.

Flujo Básico:

1. El Usuario accede a la opción de listar los tickets de su sección.
2. El Sistema presenta al Usuario una lista de todos los tickets de su sección.
3. El Sistema presenta al Usuario un formulario de asignación por cada resultado devuelto.

Flujos Alternativos:

2a. En las secciones bajo control del mánager no hay tickets abiertos:

1. El Sistema presentará un mensaje de notificación en lugar de la lista.

Caso de Uso 5: Listar Tickets de mi Sección

4.4.6 Detalles Ticket

Caso de Uso UC6: Detalles Ticket

Actor Principal: Usuario

Personal involucrado y lista de intereses:

- Cliente: Quiere ver los detalles de cualquier ticket de los que tiene abiertos.
- Técnico: Quiere ver los detalles de cualquier ticket que tenga abierto a su nombre o asignado en ese momento.
- Manager: Quiere ver todos los detalles de cualquier ticket del sistema, junto con su historial de modificaciones.

Precondiciones: El Usuario está autenticado en el sistema.
Postcondiciones: El Sistema ha mostrado al Usuario un cuadro con los detalles del Ticket solicitado.
Flujo Básico:

1. El Usuario accede a la opción de visualizar detalles.
2. El Sistema presenta al Usuario un cuadro con los detalles del ticket.

Flujos Alternativos:

1a. El usuario es de tipo Cliente:
   1. Podrá acceder a través del listado de sus tickets.

1b. El usuario es de tipo Técnico:
   1. Podrá acceder a través del listado de sus tickets (UC3), del listado de sus asignaciones (UC4), y desde el cuadro de búsqueda rápida (UC12).

1c. El usuario es de tipo Mánager:
   1. Podrá acceder a través del listado de sus tickets (UC3), del listado de tickets sin asignar de su sección (UC5), desde el cuadro de búsqueda rápida (UC12).

2a. El usuario es de tipo Cliente:
   1. Se le mostrarán los detalles del ticket: Título, descripción, fecha de apertura, técnico asignado actualmente.

2b. El usuario es de tipo Técnico:
   1. Se le mostrarán los detalles del ticket: Título, descripción, fecha de apertura, usuario creador. Se le mostrará el historial de modificaciones que ha tenido el ticket.

2c. El usuario es de tipo Mánager:
   1. Se le mostrarán los detalles del ticket: Título, descripción, fecha de apertura, usuario creador, técnico asignado, centro y sección afectados.
   2. Se le mostrará el historial de modificaciones que ha tenido el ticket.

Caso de Uso 6: Detalles Ticket

4.4.7 Asignar Técnico

Caso de Uso UC7: Asignar Técnico

Actor Principal: Mánager

Personal involucrado y lista de intereses:

- Mánager: Quiere asignar un ticket a un técnico

Precondiciones: El Usuario está autenticado en el sistema y su rol el de Mánager.
Postcondiciones: El Ticket pasa al estado "Asignado" y se le ha asociado un técnico.
Flujo Básico:

1. El Usuario accede a la opción de listar los tickets de su sección sin asignar(UC5), o bien a la opción de detalles de ticket (UC6).
2. El Usuario elige uno de los grupos de técnicos del sistema.
3. El Sistema dará la opción de seleccionar un técnico perteneciente al grupo elegido.
4. El Usuario elige uno de los técnicos disponibles.
5. El sistema notificará al Usuario con el resultado de la operación.

Flujos Alternativos:

2a. En las secciones bajo control del mánager no hay tickets abiertos:

1. El Sistema presentará un mensaje de notificación en lugar de la lista.

Caso de Uso 7: Asignar Técnico

4.4.8 Crear Usuario

Caso de Uso UC8: Crear Usuario

Actor Principal: Mánager

Personal involucrado y lista de intereses:

- Mánager: Quiere crear un nuevo usuario en el sistema.

Precondiciones: El Usuario está autenticado en el sistema y su rol es el de Mánager.
Postcondiciones: Un nuevo usuario queda registrado en el sistema.
Flujo Básico:

1. El Usuario accede a la opción de crear usuario.
2. El Sistema presenta al Usuario un formulario para la entrada de datos.
3. El Usuario introduce los datos y los envía.
4. El Sistema comprueba que los datos son válidos.
5. El Sistema almacena persistentemente el usuario, asignando un número único e identificativo.

Flujos Alternativos:

3a. Usuario tipo Cliente:

1. El Sistema solicitará el centro al que pertenece.

3b. Usuario tipo Técnico:

1. El Sistema solicitará el mánager responsable y el grupo al que pertenece.

Caso de Uso 8: Crear Usuario

4.4.9 Perfil de Usuario

Caso de Uso UC9: Modificar Perfil Usuario

Actor Principal: Usuario
Personal involucrado y lista de intereses:

- Usuario: Quiere visualizar sus datos personales y poder modificarlos

Precondiciones: El Usuario está autenticado en el sistema.
Postcondiciones: El Usuario ha modificado su perfil permanentemente.
Flujo Básico:

1. El Usuario accede a la opción de perfil de usuario.
2. El Sistema presenta al Usuario un cuadro con sus datos: Nombre, apellidos, teléfono, centro al que pertenece, rol, foto.
3. El Usuario puede modificar los datos: teléfono y foto.
4. El Sistema comprueba que los datos son válidos.
5. El Sistema almacena persistentemente el teléfono y la foto, eliminando la anterior

Flujos Alternativos:

2a. Usuario no tiene foto:

1. El Sistema muestra una foto por defecto.

Caso de Uso 9: Perfil de usuario

4.4.10 Cambiar Contraseña

Caso de Uso UC10: Cambiar Contraseña

Actor Principal: Usuario
Personal involucrado y lista de intereses:

- Usuario: Quiere modificar su contraseña

Precondiciones: El Usuario está autenticado en el sistema.
Postcondiciones: El Usuario ha modificado su contraseña de forma segura permanentemente.
Flujo Básico:

1. El Usuario accede a la opción de cambiar contraseña.
2. El Sistema presenta al Usuario dos campos para introducir la nueva contraseña
3. El Usuario introduce la nueva contraseña.
4. El Sistema almacena la nueva contraseña encriptada en la base de datos.

Flujos Alternativos:

3a. El contenido de los dos campos no coincide

1. Se notifica al usuario de que las dos contraseñas deben coincidir

Caso de Uso 10: Cambiar contraseña

4.4.11 Buscar Ticket

Caso de Uso UC11: Buscar Ticket por Número

Actor Principal: Técnico o mánager
Personal involucrado y lista de intereses:

- Usuario: Quiere buscar un ticket de los disponibles en la base de datos a partir de su número

Precondiciones: El Usuario está autenticado en el sistema.
Postcondiciones: El Usuario ha buscado un ticket por su número identificativo.
Flujo Básico:

1. El Usuario escribe el número de ticket en un campo de texto visible en todo momento.
2. El Sistema presenta al Usuario los detalles del ticket en cuestión.

Flujos Alternativos:

2a. El ticket no existe en el sistema

1. Se notifica al usuario de que el ticket no se ha encontrado.

Caso de Uso 11: Buscar ticket por Número

4.4.12 Cerrar sesión

Caso de Uso UC11: Cerrar sesión

Actor Principal: Usuario
Personal involucrado y lista de intereses:

- Usuario: Quiere cerrar su sesión de usuario

Precondiciones: El Usuario está autenticado en el sistema.
Postcondiciones: El Usuario no está autenticado en el sistema.
Flujo Básico:

1. El Usuario accede a la opción *Cerrar sesión.*
2. El Sistema elimina la sesión actual y muestra la pantalla de login.

Flujos Alternativos:

1a. El usuario permanece inactivo durante 10 minutos

1. El Sistema elimina la sesión actual y muestra la pantalla de login.

Caso de Uso 12: Cerrar sesión

5 BLOQUE V:
DISEÑO

En esta parte del documento se va a detallar el proceso de diseño de la arquitectura del sistema. En la primera parte de la sección se elaborará un diseño arquitectónico del sistema, incluyendo aspectos organizacionales, una subdivisión del sistema completo en subsistemas lógicos que ayudarán a comprender qué elementos físicos se necesitan para componer el sistema. Se irán aplicando patrones de diferentes referencias que ayudarán a diseñar el sistema de modo que cumpla con las condiciones de calidad de software de forma probada y documentada.

5.1 DISEÑO ARQUITECTÓNICO

En esta y en las sucesivas secciones se hará un diseño arquitectónico del sistema UVaHelpDesk obteniendo una serie de subsistemas que compondrán el sistema completo, junto con las relaciones entre los mismos.

El diseño debe ser realizado de forma que se cumplan las siguientes condiciones de manera que haya un equilibrio entre todas ellas:

- *Rendimiento y mantenimiento*: Se intentará hacer que el rendimiento y el mantenimiento del sistema estén en equilibrio de forma que los componentes sean del suficiente grano grueso como para ofrecer un buen rendimiento y al mismo tiempo el suficiente grano fino como para que un cambio o una ampliación del sistema produzca los menores trastornos posibles.
- *Protección*: Se protegerán las partes más sensibles del sistema (los datos) en una capa interna añadiendo niveles de protección para llegar a ella.
- *Seguridad*: El sistema ha de asegurar con el mayor grado posible la confidencialidad, integridad y disponibilidad de los datos.
- *Disponibilidad*: el diseño estará orientado a la posible replicación de ciertos componentes tales como los servidores de BD y Web en un futuro.

5.1.1 Organización del sistema

Para la organización más elemental del sistema se utilizará uno de los modelos más extendidos: el modelo **cliente-servidor** [21]. Es el modelo más adecuado para un sistema de estas características, en el que sus elementos se interrelacionan a través de internet (protocolo HTTP), así, este modelo tiene la ventaja de ser una arquitectura distribuida, es posible hacer una separación de los elementos diferenciados más sencilla, mantenible y segura. De esta forma ya se diferencian 2 grandes subsistemas: los clientes, y el servidor.

Se intentará integrar el modelo cliente-servidor con un modelo de **repositorio de datos** [21], ya que todos los datos compartidos se almacenan en una base de datos central a la cual pueden acceder todos los subsistemas directa, o indirectamente.

También se aplicará un **modelo de capas** [21] a todo el sistema separando la capa de datos, la capa de aplicación y la capa de presentación. De esta forma, cada capa sólo puede comunicarse con sus capas adyacentes añadiéndose un nivel de protección extra al estar los datos en la capa más interna de la aplicación. Así tendremos un modelo cliente-servidor de tres capas con repositorio de datos. De esta forma, el servidor queda descompuesto en otros dos subsistemas, uno que se encargará de la lógica de los datos y el otro de la lógica de aplicación.

Modelo de cliente: Se entiende por cliente a cada uno de los terminales que se conecten al servicio por medio de internet, en estos se delega la menor responsabilidad posible, limitándose a implementar la lógica de presentación por medio del navegador web.

Quedan identificados 3 grandes subsistemas que formarán el sistema completo de una forma lógica, que se asociarán a cada una de las capas del modelo cliente-servidor de tres capas:

- **Lógica de presentación**: Hará las funciones de intermediario entre los usuarios y el sistema permitiendo el intercambio de información entre ambos. Tiene que ser capaz de mostrar los datos a los usuarios y recoger consultas de los mismos para su procesamiento en la siguiente capa.

- **Lógica de negocio**: Se encargará de toda la gestión interna del programa. Debe ser capaz de obtener datos de la capa de presentación y llevar a cabo la gestión de todo lo referente al procesamiento de los datos para que se cumplan los objetivos de la especificación.
- **Persistencia**: Este subsistema será el asociado a la última capa del modelo, contendrá todos los datos que necesita la aplicación para su funcionamiento acorde a las especificaciones.

5.1.2 Flujo de Control

Para la parte del sistema encargada de llevar la gestión de las transacciones realizadas desde el portal web se utilizará un modelo de control centralizado basado en un **modelo del gestor** [3] al tratarse de un sistema concurrente.

5.1.3 Diseño físico de la arquitectura

Con el análisis a nivel conceptual obtenido de las anteriores secciones, ya se dispone de un punto de partida más cercano a la solución del problema con una serie de elementos bien diferenciados y definidos y sus interrelaciones. Se intentará por medio de la tecnología disponible ajustarse de la forma más precisa al problema.

En el diagrama de despliegue de la Figura 18, se muestra la propuesta para los diferentes dispositivos que conformarán el sistema y los protocolos que se utilizarán para comunicarlos. Se contará con dos elementos: el cliente, con un navegar web cualquiera instalado, y el servidor que contendrá tanto el servidor web con el procesador de PHP, los ficheros y los elementos multimedia, como el SGBD (Sistema Gestor de Bases de Datos) con la base de datos correspondiente.

5.2 DECISIONES DE DISEÑO

Al tratarse de una aplicación que requiere una alta interactividad con el usuario, se ve conveniente aplicar el patrón POSA **Modelo-Vista-Controlador** [21], separando la funcionalidad y los datos (Modelo), de los elementos interactivos (Vista), y del control entre estos dos elementos (Controlador), recibiendo este último eventos del usuario y enviando las solicitudes al modelo y viceversa.

Para las distintas capas identificadas del sistema, se van a utilizar algunos de los patrones propuestos por Fowler [22] en *"Patterns of Enterprise Application Architecture"* divididos en tres funcionalidades fundamentales:

- **Lógica de dominio**: como el sistema está estructurado de una manera orientada a objetos se hará uso de **Domain Model** [22] ya que requiere una representación afín al modelo de dominio identificado en la fase de análisis y de la misma forma que sea también afín a las tablas de la base de datos. Por tanto, por cada una de las entidades identificadas se elaborará una clase, así como una tabla en la base de datos en la medida de lo posible.
- **Fuente de datos**: dado que la estructura de la base de datos es similar al modelo de dominio se utilizará **Data Mapper** [22] para establecer una capa que separe los objetos del dominio de la base de datos. Su responsabilidad será transferir los datos entre dichas partes de forma que estén aisladas entre ellas, encargándose el Data Mapper de realizar las consultas a la base de datos de forma transparente para el modelo. Por tanto, cada objeto del dominio tendrá su correspondiente Mapper con las funciones de acceso a la base de datos necesarias. De esta forma se encapsula toda la parte de SQL en este DataMapper facilitando la asignación de responsabilidades a cada uno de los elementos que conforman el sistema.
- **Presentación**: Para la parte de presentación se hará uso de **Front Controller** [22] para recibir las peticiones del usuario e invocar al controlador correspondiente tras analizar la petición, todas las peticiones pasarán previamente por el Front Controller. La presentación de datos se hará por medio de **Template View** [22], un fichero HTML con código PHP incrustado para presentar los datos que se le pasen desde el controlador.

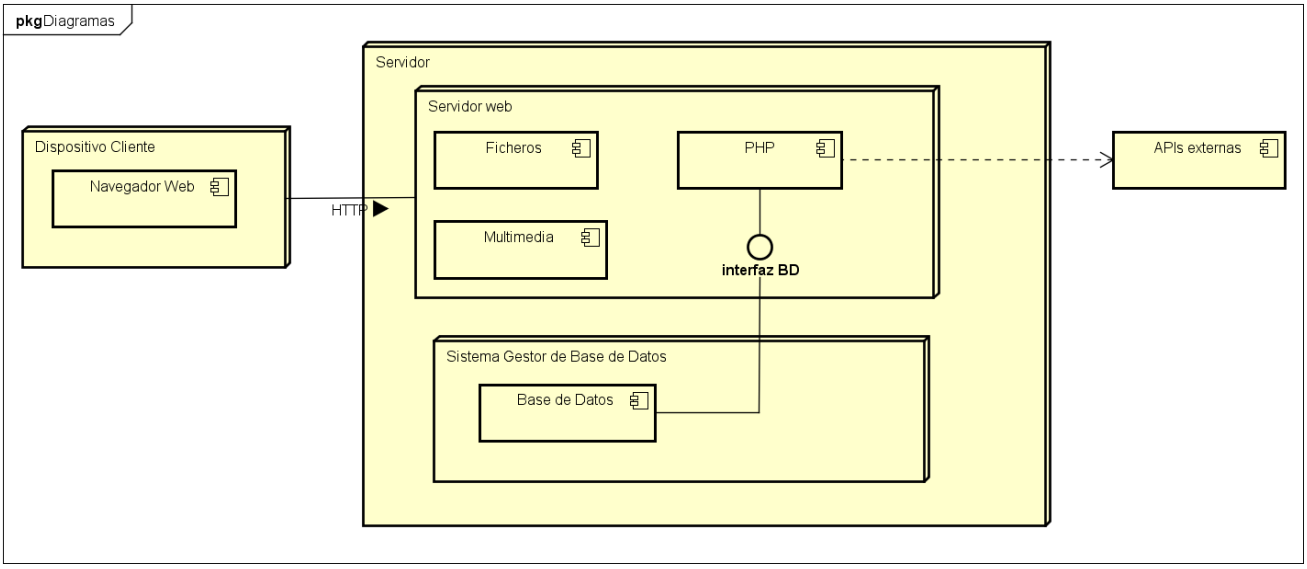


Figura 18: Arquitectura del Sistema

Como se ha justificado en la sección 2.4 Se ha escogido PHP como lenguaje de programación en la parte de servidor por su gran acogimiento en la comunidad de desarrolladores, su flexibilidad y las opciones de integración que ofrece con otras tecnologías (Bases de datos, Bootstrap, JavaScript y jQuery, AJAX). Se trata de un lenguaje muy asentado y así se facilitará el futuro desarrollo o extensión de la aplicación, teniendo un modelo debidamente documentado e implementado.

Por tanto, se obtienen las siguientes partes interrelacionadas entre sí. A continuación, se explican las responsabilidades de cada una de ellas:

- **Vista**: La vista engloba toda la parte de la aplicación cuya responsabilidad es la de presentar la información al usuario que la esté utilizando. Así mismo se encargará de recoger los datos que este introduce por medio de formularios o clicks en enlaces. Las únicas funcionalidades integradas en las vistas serán scripts del lado del cliente que sean imprescindibles para realizar ciertas funcionalidades mínimas definidas en los requisitos, como el intercambio asíncrono de datos por medio de AJAX, o la primera validación, antes de que los datos se envíen al controlador (esto serán validaciones muy básicas: alfanuméricas, introducción de emails, comprobar que todos los campos del formulario están cumplimentados…). La vista nunca se comunicará directamente con el modelo de dominio, siempre tendrá que existir un controlador como intermediario.
- **Controlador**: Esta parte del diseño será la encargada de decidir qué funciones se ejecutan en función de los datos recibidos por la vista. Como se ha dicho existirá un controlador frontal que reciba todas las peticiones y que delegue en el resto de controladores especializados las tareas a realizar. Esta idea se ha obtenido del conjunto de patrones GoF (Gang of Four) [23], en concreto del patrón Comando, así como del patrón Front Controller definido en *Patterns of Enterprise Application Architecture* [22].
  Cada uno de los controladores realizará la labor de recoger los datos que se han enviado vía HTTP desde la vista, llamar a las funciones necesarias del modelo de dominio, y volver a ejecutar una vista, junto con los datos que esta necesitará para mostrar la información necesaria. Las vistas tan sólo podrán ser llamadas desde los controladores.
- **Modelo de Dominio**: En el modelo de dominio estará englobada la funcionalidad propiamente dicha de la aplicación, así como sus características. El modelo de dominio será utilizado por medio de los controladores. Así se facilita la integración con otros sistemas al no depender directamente de la vista. Este modelo podrá ser utilizado por otros controladores, aunque el ámbito sea totalmente diferente, ya que no depende de ningún modo de ellos ni de las vistas.
- **DataMapper**: Para evitar mezclar código SQL con el modelo de dominio se ha optado por aplicar una capa de interacción con la base de datos desde el modelo. De esta forma se separarán totalmente estos dos elementos de manera que el acceso a los datos sea totalmente transparente para el

modelo. El modelo le pedirá los datos al DataMapper, y éste se encargará de hacer la consulta apropiada al SGDB (Sistema Gestor de Bases de Datos), y posteriormente los devolverá al modelo de una forma consistente con el mismo. Esto facilitará integraciones del modelo con otros SGDB, incluso con filosofías de acceso a datos diferentes a las bases de datos de tablas relacionales (bases de datos orientadas a objetos, bases de datos de red, transaccionales…). Bastaría con implementar un nuevo DataMapper sin necesidad de modificar el código del Modelo de Dominio.
Existen librerías para PHP con funcionalidades propias de un DataMapper. Estas suelen proporcionar las clásicas operaciones CRUD (Crear, Leer, Actualizar y Borrar) que se ejecutan en una base de datos. No obstante, se ha optado por la implementación del DataMapper para tener un mayor control sobre las operaciones que se hagan a la base de datos y sean más precisas y óptimas.

5.3 MODELOS DEL SISTEMA

Dadas las características anteriormente estudiadas, el resultado del proceso de diseño se materializa en una serie de modelos del sistema, tanto estáticos como dinámicos. En la parte de modelos estáticos se mostrarán una serie de diagramas que reflejan el modelo del sistema, en lo que a su construcción respecta. Los modelos dinámicos reflejarán el comportamiento de las entidades definidas para el modelo estático entre ellas, sus relaciones en tiempo de ejecución a la hora de llevar a cabo la funcionalidad requerida.

5.3.1 Modelos estáticos

Se ha decidido separar los diagramas de forma que se pueda ver en detalle cada uno de los componentes que forman la aplicación ya que el diagrama al completo puede resultar demasiado extenso para el formato de este documento. Aun así, se adjunta el diagrama de clases completo junto con el resto de modelos en el CD.

5.3.1.1 Modelo de Dominio

Los elementos identificados quedan reflejados en la Figura 19. A continuación, se detalla la función que cumple cada uno de los elementos identificados.

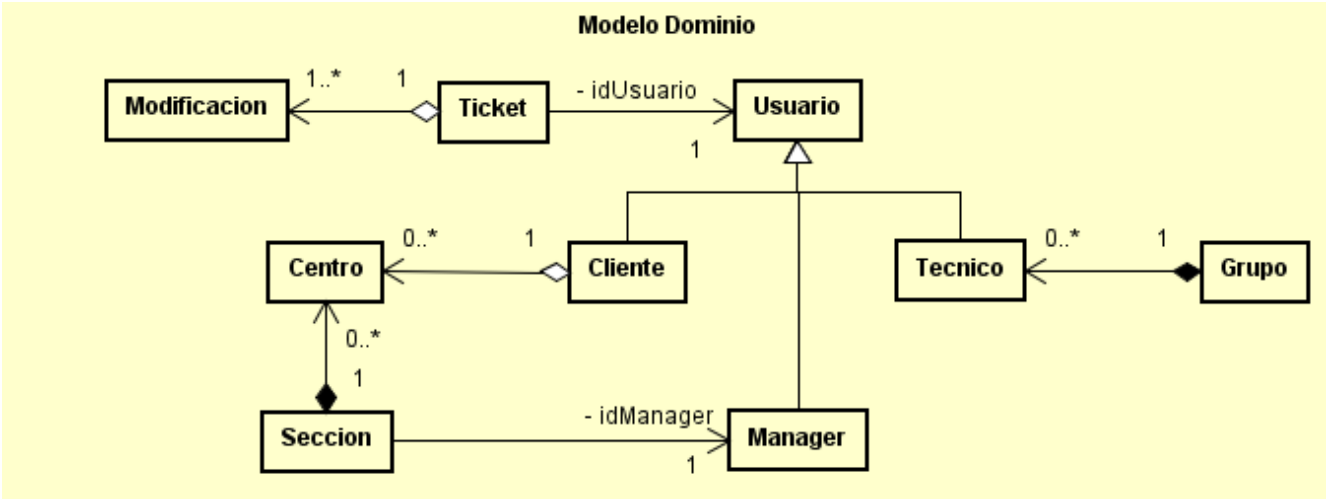


Figura 19: Modelo de Dominio

5.3.1.1.1 Ticket

Representa una incidencia con una estructura fiel a la que se sigue en la base de datos y encapsula toda la funcionalidad relativa al tratamiento de información referente a las incidencias.

Cualquier usuario puede crear sus propios tickets (a su nombre) y cada uno de los tickets llevará asociado un conjunto de modificaciones que se han hecho sobre el mismo para así poder mantener un historial de modificaciones y poder ver la evolución del mismo a lo largo del tiempo.

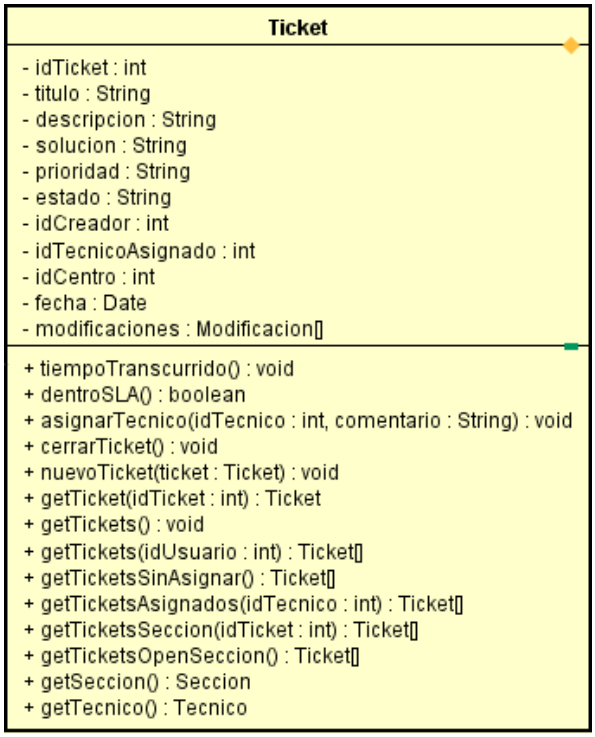


Figura 20: Ticket

5.3.1.1.2 Usuario

Clase de la que derivarán los tipos de usuario especializados: Técnico, Cliente y Mánager. En esta clase se localizarán las funciones comunes a todos los tipos de usuario como pueden ser el login o cambiar la foto de perfil. También se guardarán los campos comunes a todos los tipos de usuario: idUsuario, nombre, apellidos, tipo, email, contraseña, teléfono y foto.

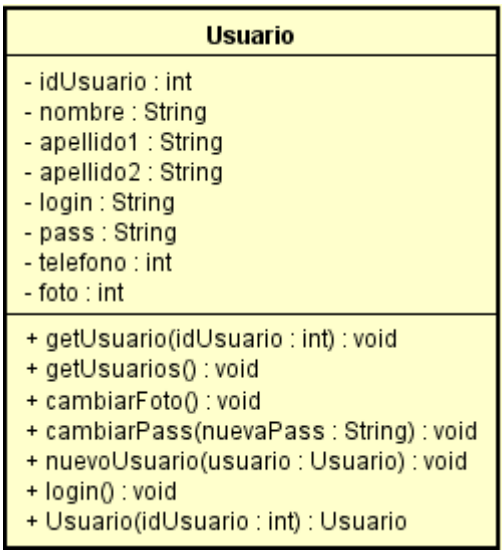


Figura 21: Usuario

5.3.1.1.3 Técnico

Clase especializada a partir de la clase Usuario, contendrá los datos exclusivos de un técnico: idManager, que representa el identificador del mánager supervisor del técnico en cuestión, e idGrupo, que contendrá el identificador del grupo al que pertenece el técnico.

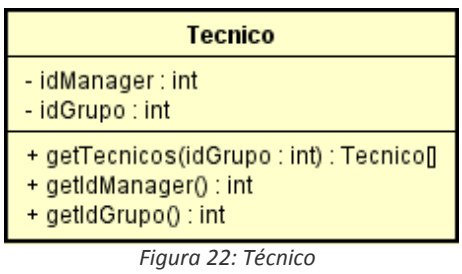


Figura 22: Técnico

5.3.1.1.4 Cliente

Clase especializada a partir de la clase Usuario, contendrá los datos exclusivos de un cliente: idCentro, que representará el identificador del centro al que pertenece el cliente en cuestión.

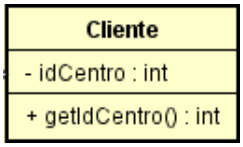


Figura 23: Cliente

5.3.1.1.5 Mánager

Clase especializada a partir de la clase Usuario, contendrá los datos exclusivos de un mánager.

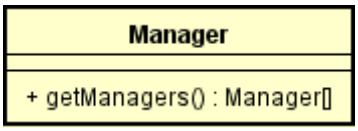


Figura 24: Mánager

5.3.1.1.6 Centro

Representa cada uno de los centros con los que se trabajará, de una forma similar a la que existe en la tabla de la base de datos. Se guardarán los datos definidos para los centros e implementará las funciones que se realizarán para los mismos.

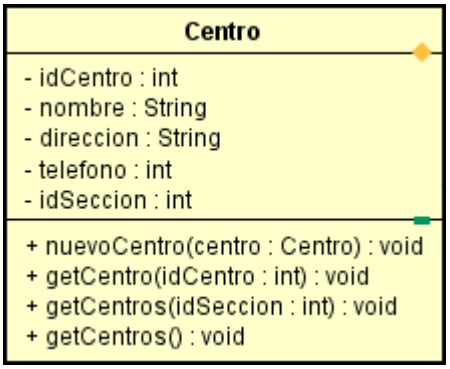


Figura 25: Centro

5.3.1.1.7 Grupo

Contendrá los campos y operaciones definidas para los grupos de técnicos que pueden atender los tickets que contiene la aplicación.

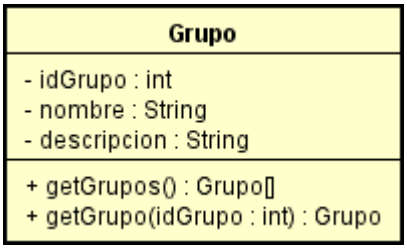


Figura 26: Grupo

5.3.1.1.8 Sección

Contendrá los campos y operaciones definidas para las secciones que agrupan centros de los que constará la aplicación.

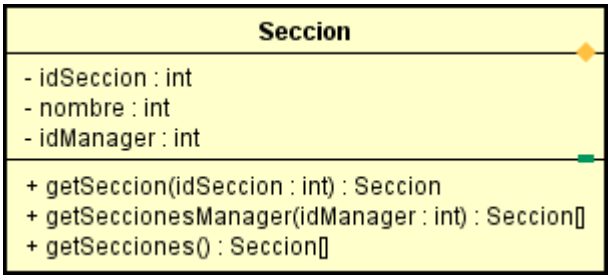


Figura 27: Sección

5.3.1.2 Data Mapper

La capa de DataMapper encapsulará todo lo relativo al tratamiento con la base de datos. Todo el código SQL debe ir en este lugar sin excepción. De esta forma se añade un grado de mantenibilidad a la aplicación: no se mezclará código SQL con el código del modelo de dominio, de esta forma un experto en bases de datos sabrá donde tiene que editar la aplicación para cumplir su función. El DataMapper realizará listados, inserciones y modificaciones. Devolviendo NULL en caso de que la consulta no devuelva ningún resultado.

5.3.1.2.1 DataMapperTicket

Se encarga de realizar las consultas que devuelvan datos de tipo Ticket, o bien actualicen las tablas de la base de datos correspondientes. Estas consultas incluyen listados de tickets con una determinada característica (creados por un determinado usuario, pertenecientes a una determinada sección, con un estado concreto), así como la inserción de nuevos tickets con los datos introducidos por el usuario en la vista; o modificaciones, englobando los cambios de estado que puede sufrir un ticket a lo largo de su ciclo de vida por medio de la interacción con el usuario.

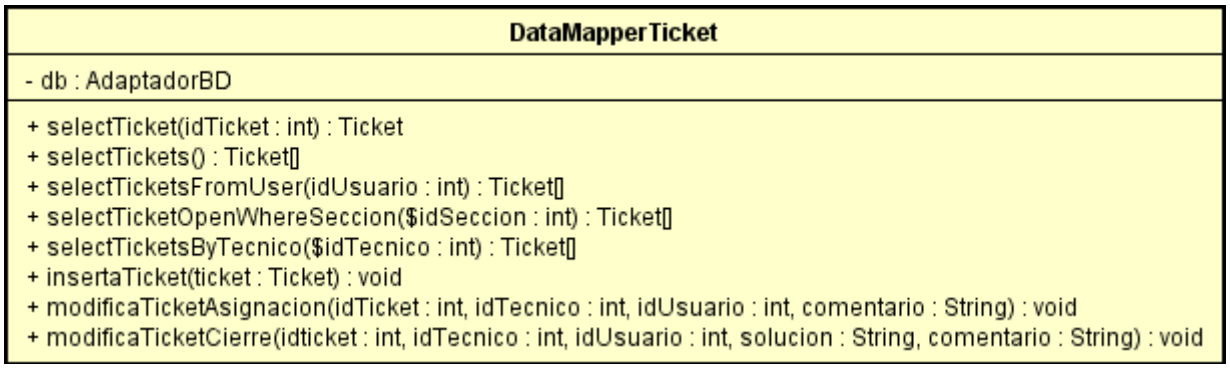


Figura 28: DataMapper Ticket

5.3.1.2.2 DataMapperUsuario

Se encarga de realizar las consultas que devuelvan datos de tipo Usuario (incluyendo técnicos, clientes y mánagers), o bien actualicen las tablas de la base de datos necesarias. Estas consultas incluyen listados de usuarios con una determinada característica (seleccionar un usuario por su ID o email, listados de diferentes tipos de usuarios…), así como la inserción de nuevos usuarios con los datos aportados previamente por el usuario. También se contemplarán modificaciones como el cambio de la foto de perfil de un usuario, o la modificación de la contraseña.

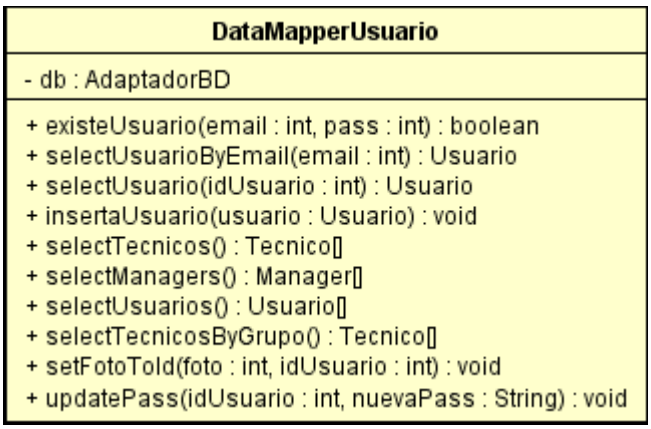


Figura 29: DataMapper Usuario

5.3.1.2.3 DataMapperCentro

Se encarga de realizar las consultas que devuelvan datos de tipo Centro, o bien actualicen las tablas de la base de datos correspondientes.

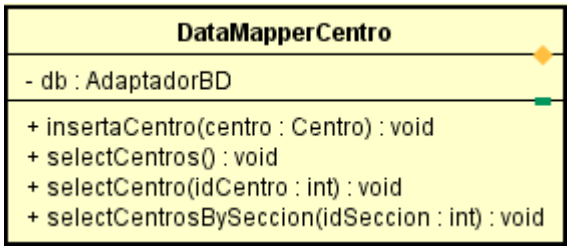


Figura 30: DataMapper Centro

5.3.1.2.4 DataMapperSeccion

Se encarga de realizar las consultas que devuelvan datos de tipo Seccion, o bien actualicen las tablas de la base de datos necesarias.

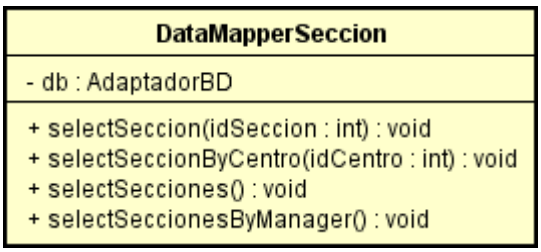


Figura 31: DataMapper Seccion

5.3.1.2.5 DataMapperModificacion

Se encarga de realizar las consultas que devuelvan datos de tipo Modificacion, o bien actualicen las tablas de la base de datos necesarias.

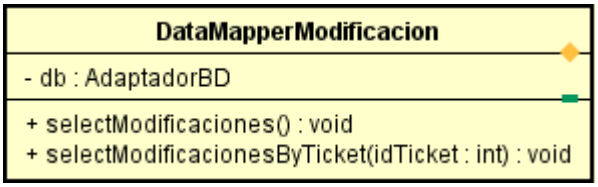


Figura 32: DataMapper Modificación

5.3.1.3 Controladores

Se encargan de recibir las peticiones del usuario desde las vistas, procesar las solicitudes recogiendo los datos enviados y pasándoselos al modelo de dominio. Una vez se reciben los datos necesarios que se han procesado en el modelo, se volverá a invocar a la vista correspondiente, mostrando los datos necesarios. En la Figura 33 se muestran los diferentes controladores.

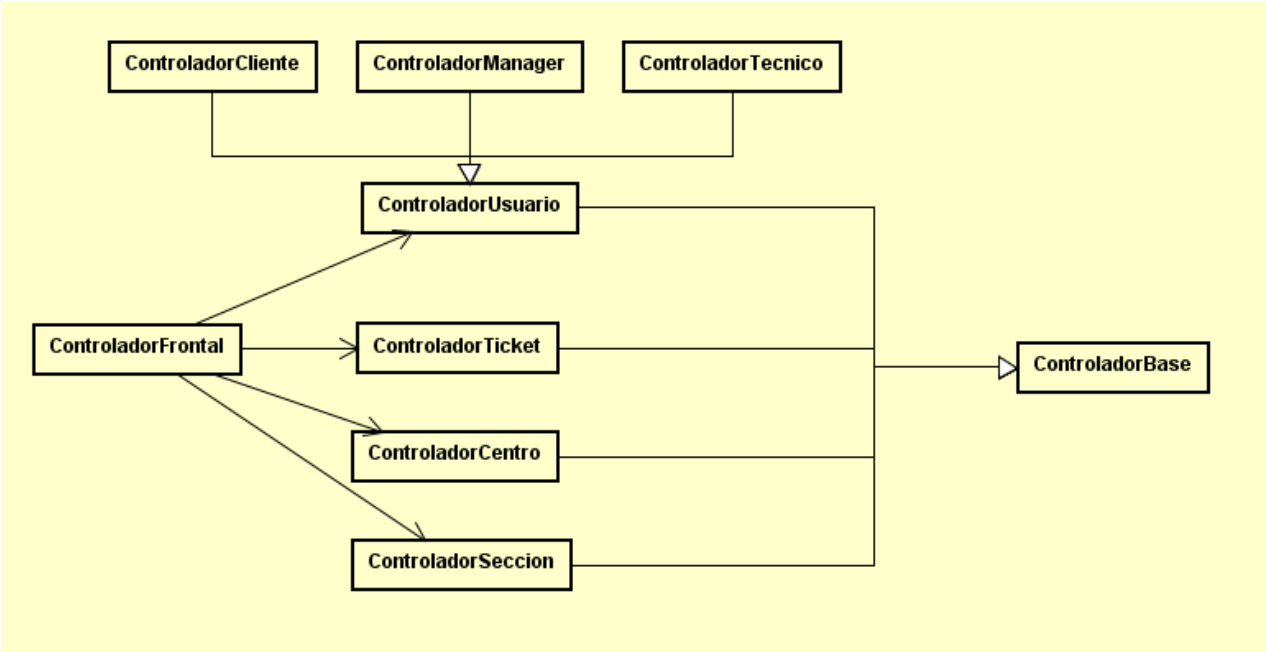


Figura 33: Controladores

5.3.1.3.1 Controlador Base

Controlador del que heredan el resto de controladores de la aplicación. Tiene como único método showPaginaVacia.

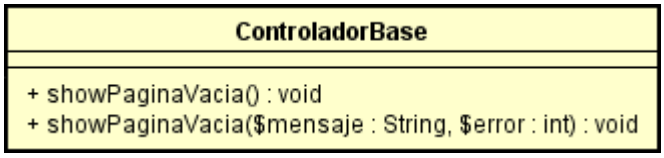


Figura 34: Controlador Base

5.3.1.3.2 ControladorFrontal

Se encargará de recibir todas las peticiones vía HTTP y llamará al controlador correspondiente dependiendo de la petición.

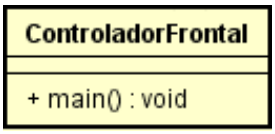


Figura 35: Controlador Frontal

5.3.1.3.3 ControladorUsuario

Lleva a cabo el control de la ejecución de los métodos del modelo de dominio (Usuario) y envía a la vista (Template View) los datos necesarios para la presentación.

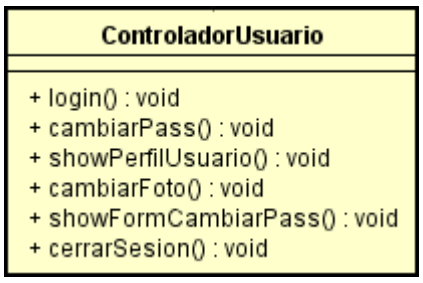


Figura 36: Controlador Usuario

5.3.1.3.4 ControladorTicket

Lleva a cabo el control de las acciones propias de la parte del modelo de dominio que maneja las incidencias y envía los resultados de vuelta a la capa de presentación.

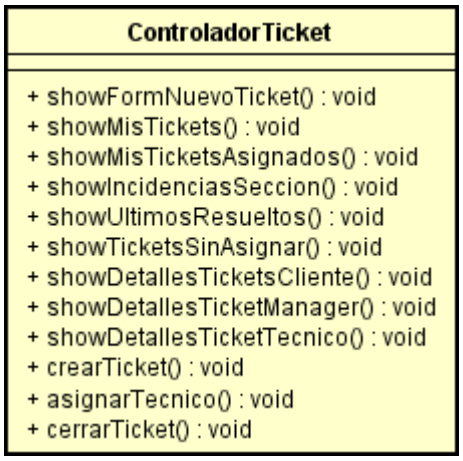


Figura 37: Controlador Ticket

5.3.1.3.5 ControladorTecnico

Lleva a cabo el control de las acciones relacionadas con la gestión de usuarios que se pueden realizar desde la interfaz de técnicos.

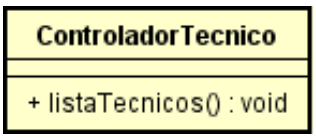


Figura 38: Controlador Tecnico

5.3.1.3.6 ControladorManager

Lleva a cabo el control de las acciones relacionadas con la gestión de usuarios que se pueden realizar desde la interfaz de técnicos.

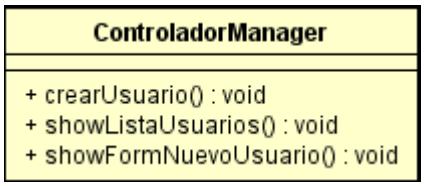


Figura 39: Controlador Manager

5.3.1.3.7 ControladorSeccion

Lleva a cabo el control de las acciones propias de la parte del modelo de dominio que maneja las secciones.

Figura 40: Controlador Sección

5.3.1.3.8 ControladorCentro

Lleva a cabo el control de las acciones propias de la parte del modelo de dominio que maneja los centros.

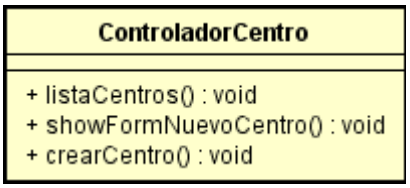


Figura 41: Controlador Centro

5.3.1.4 Vistas

Como se ha dicho se hará uso del patrón de diseño Template View para la capa de presentación de datos al usuario, así como la entrada de datos hacia la aplicación por parte del usuario.

Como se explicó en la sección 2.8, se hará uso del framework Bootstrap [9]. Este framework facilita la construcción de las vistas ofreciendo plantillas con elementos básicos que puede contener una página web (formularios, tablas, paneles, menús) combinando tecnologías HTML 5, JavaScript, jQuery y CSS. La elección de Bootstrap como framework de presentación de las vistas es debido a su facilidad de uso e integración, así como

la excelente capacidad de responsiveness que ofrece en los dispositivos móviles, esto es, la capacidad de adaptarse al tamaño de la pantalla y la elegancia de sus plantillas de libre descarga.

5.3.2 Modelos dinámicos

En este apartado se describe el sistema desde una perspectiva dinámica, es decir, se mostrará qué secuencia de mensajes desatan las peticiones de los actores las clases que lo forman para dar la respuesta adecuada a cada una de estas peticiones

A continuación, se verán los diagramas de secuencia de los casos de uso principales de la aplicación. En estos diagramas se muestra como se comunican o relacionan las clases para construir una respuesta ante una petición por parte del usuario, y se aprecia visualmente el resultado de la arquitectura software implementada, así como los patrones de diseño utilizados.

5.3.2.1 Diagrama de secuencia: Login

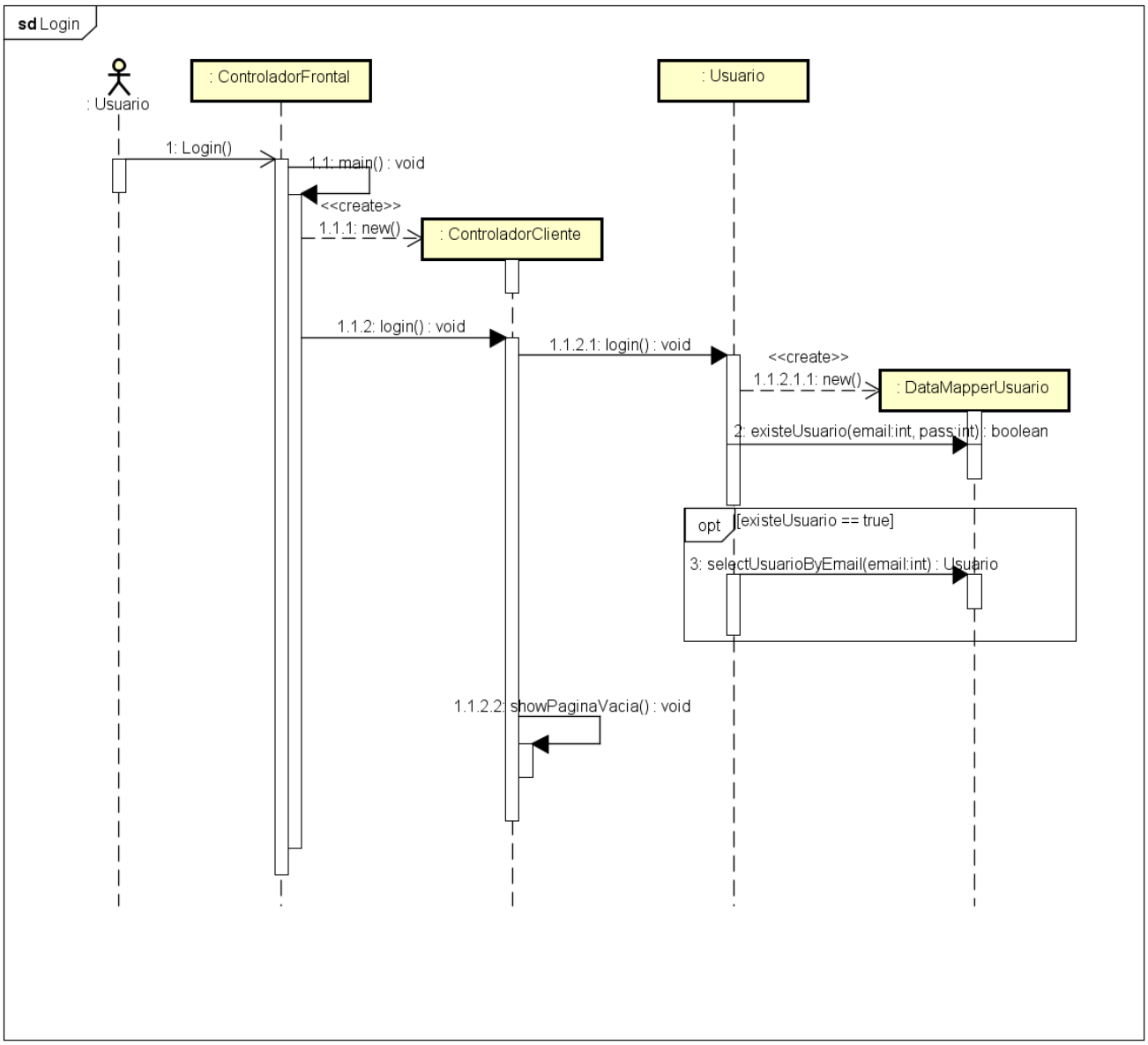


Figura 42: Diagrama de secuencia - Login

5.3.2.2 Diagrama de secuencia: Abrir Ticket

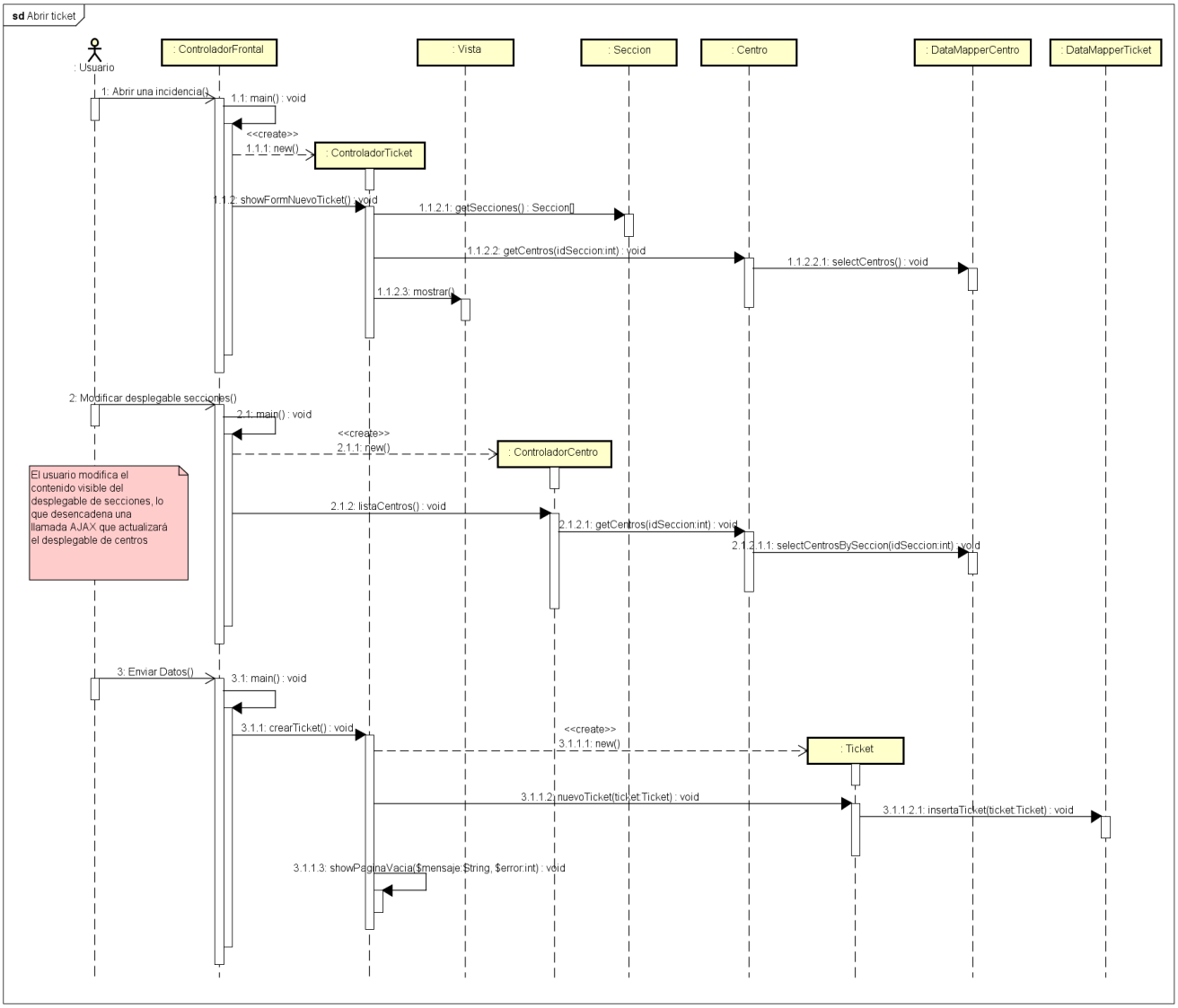


Figura 43: Diagrama de secuencia - Abrir ticket

5.3.2.3 Diagrama de secuencia: Asignar técnico

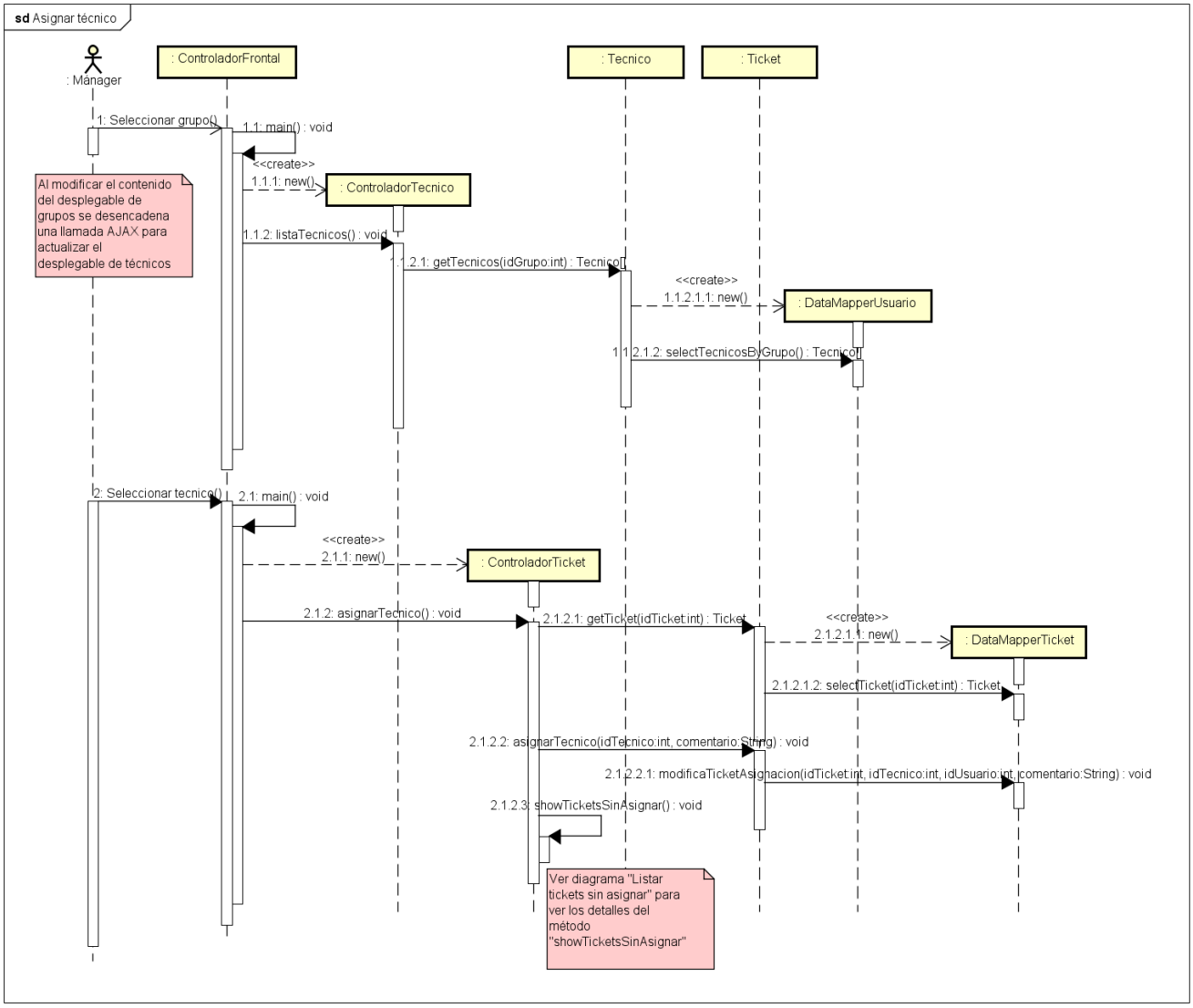


Figura 44: Diagrama de secuencia - Asignar técnico

5.3.2.4 Diagrama de secuencia: Cambiar contraseña

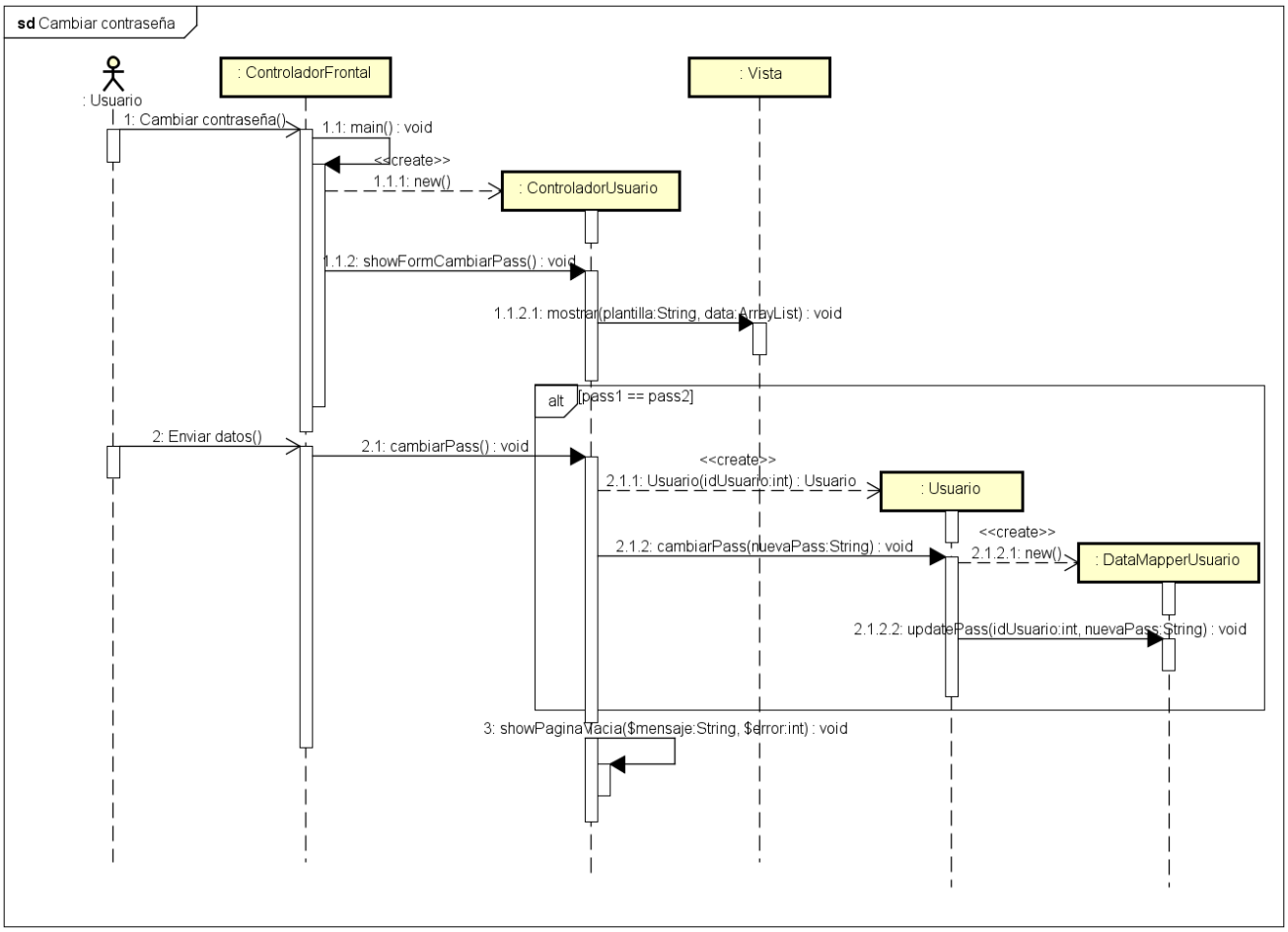


Figura 45: Diagrama de secuencia - Cmabiar contraseña

5.3.2.5 Diagrama de secuencia: Cerrar sesión

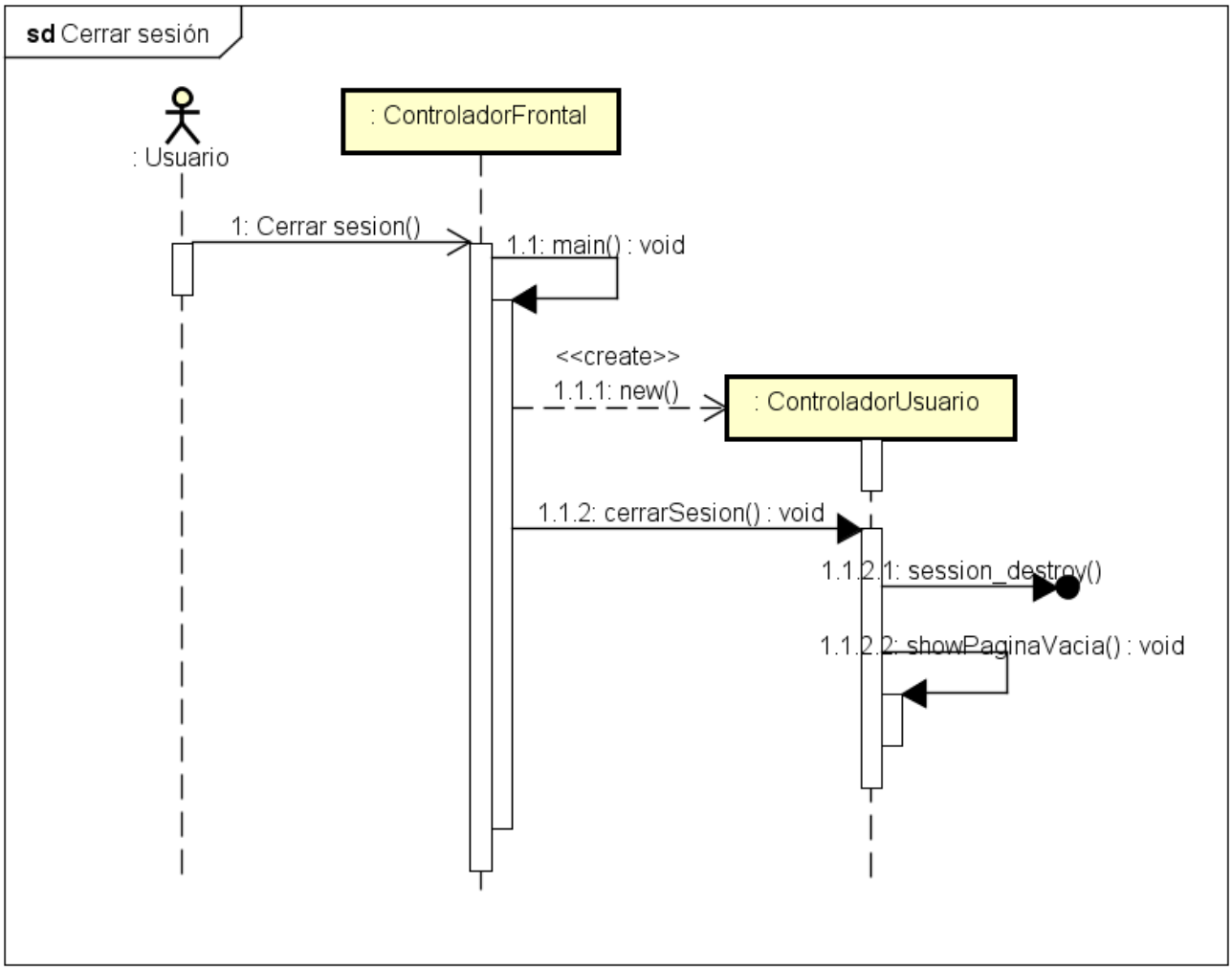


Figura 46: Diagrama de secuencia - Cerrar sesión

5.3.2.6 Diagrama de secuencia: Crear usuario

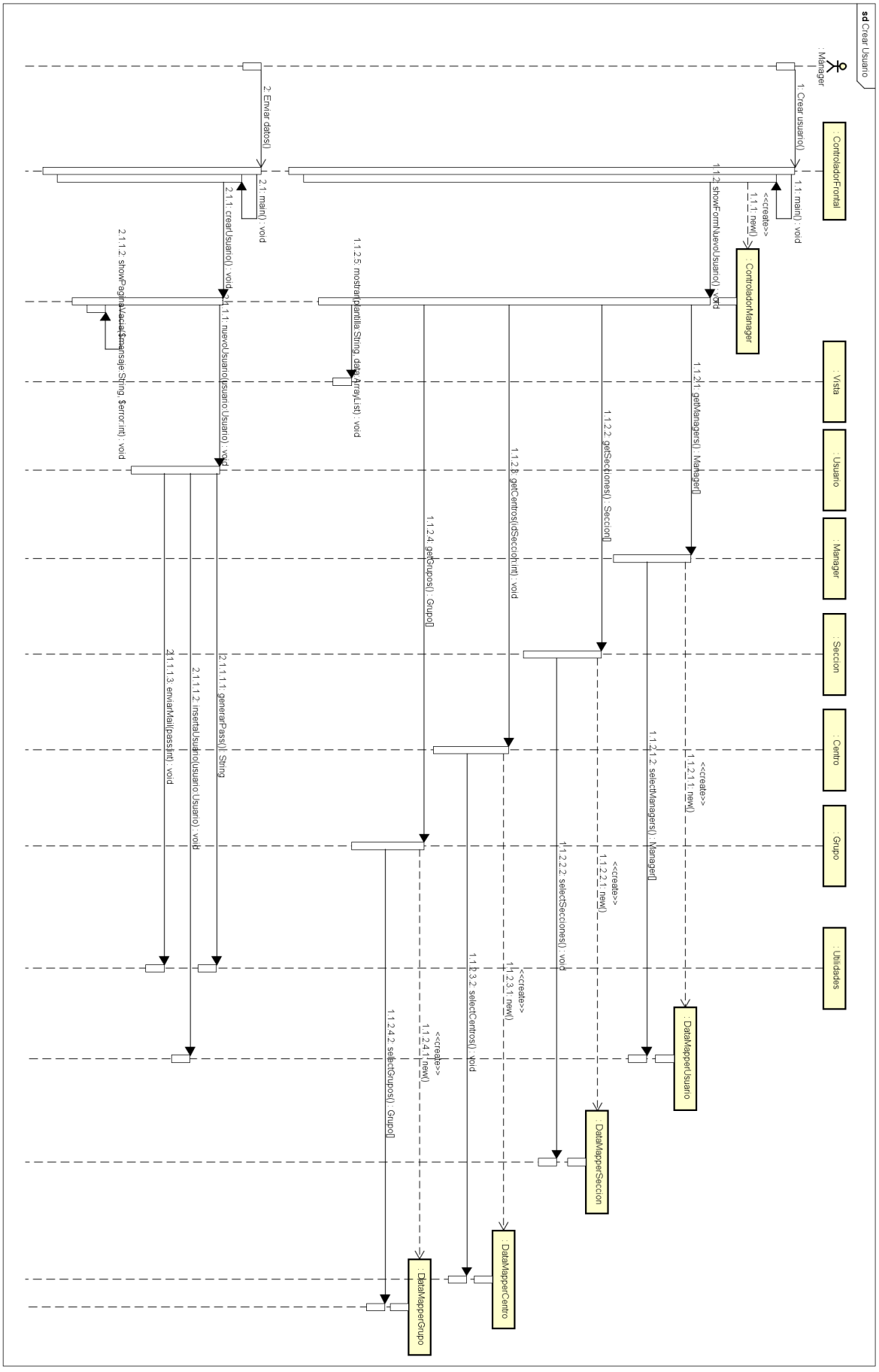


Figura 47: diagrama de secuencia – Crear usuario

5.3.2.7 Diagrama de secuencia: Detalles Ticket Manager

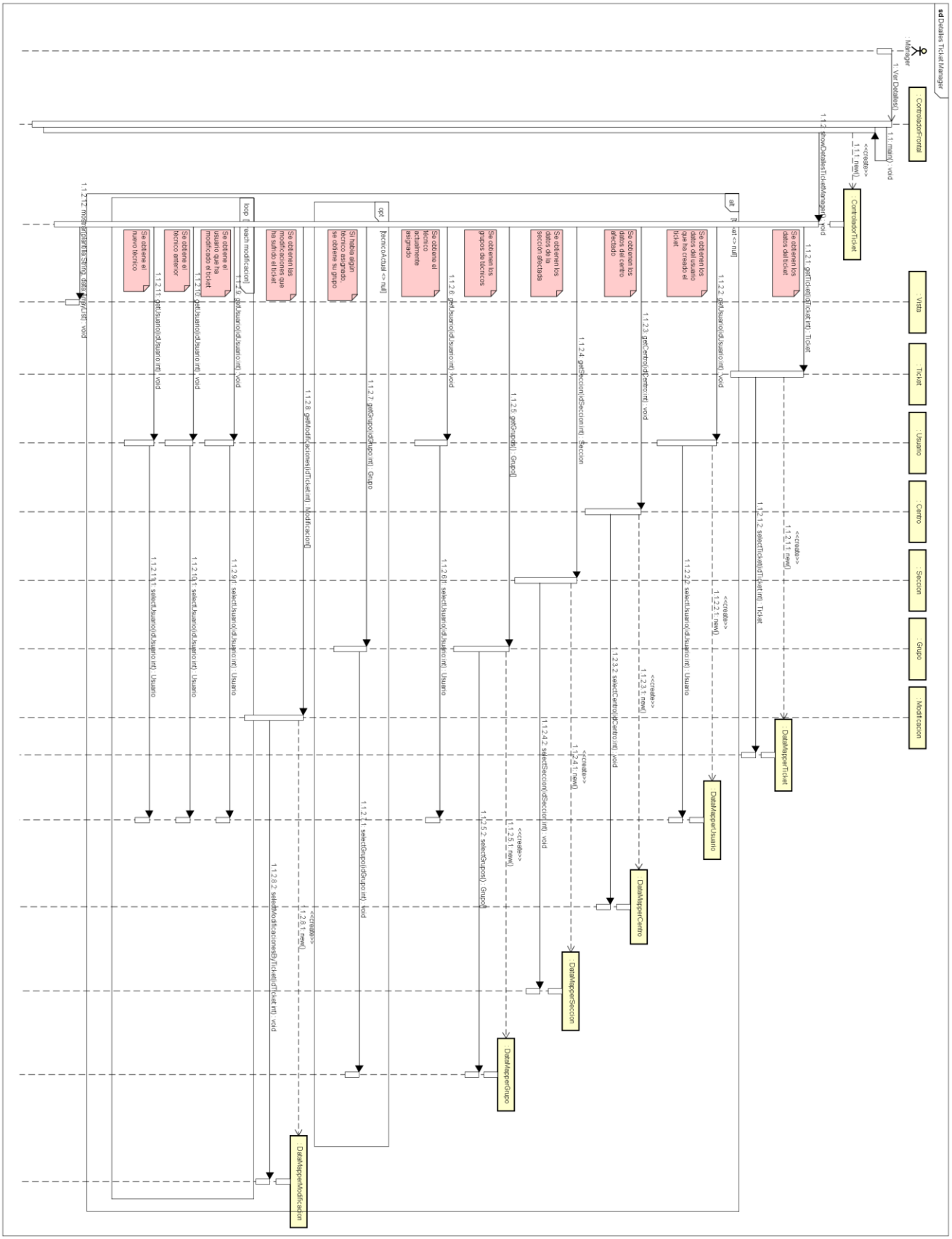


Figura 48: Diagrama de secuencia - Detalles ticket mánager

5.3.2.8 Diagrama de secuencia: Listar incidencias de mi sección

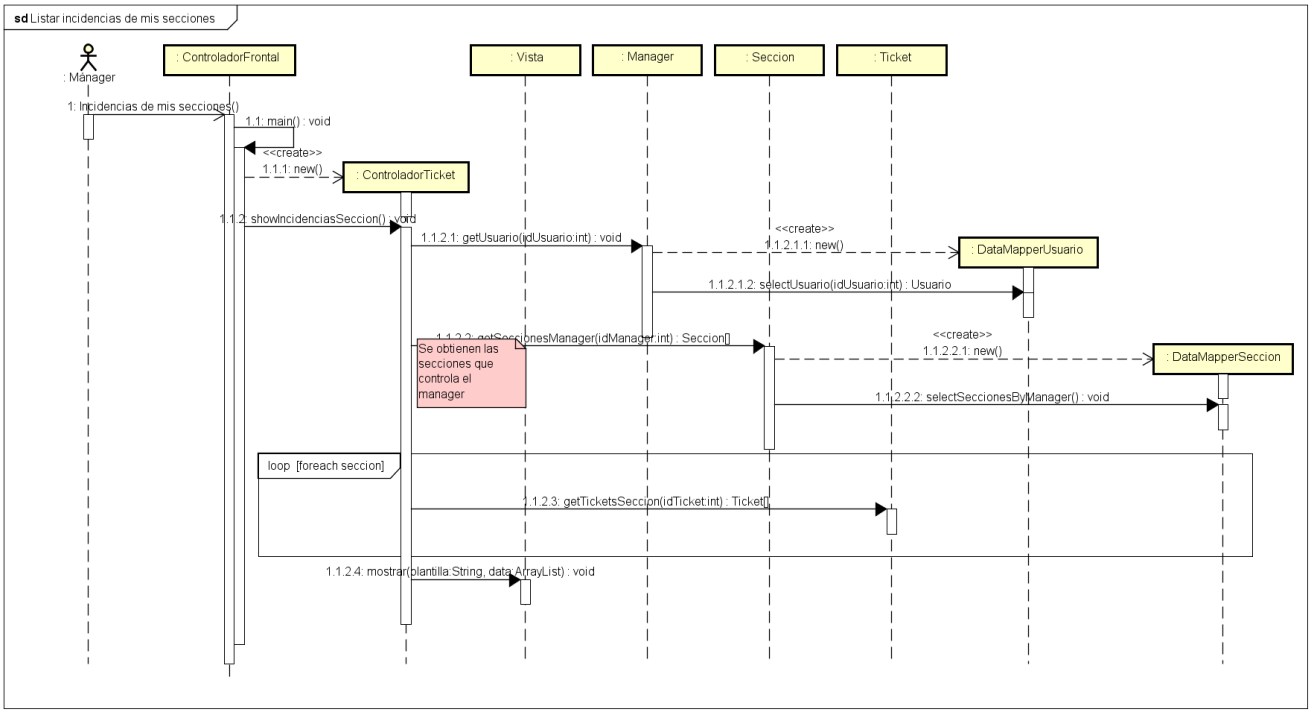


Figura 49:Diagrama de secuencia - Listar incidencias de mi sección

5.3.2.9 *Diagrama de secuencia: Listar tickets a mi nombre*

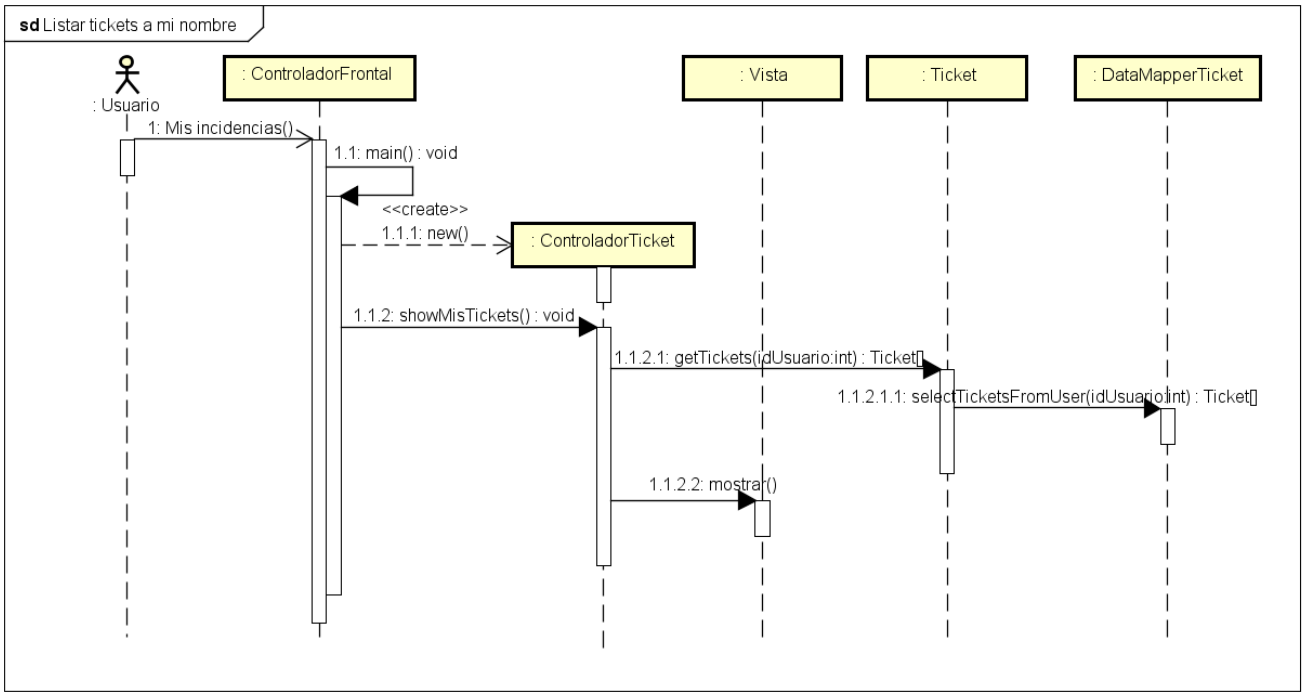


Figura 50: Diagrama de secuencia - Listar tickets a mi nombre

5.3.2.10 Diagrama de secuencia: Listar tickets asignados

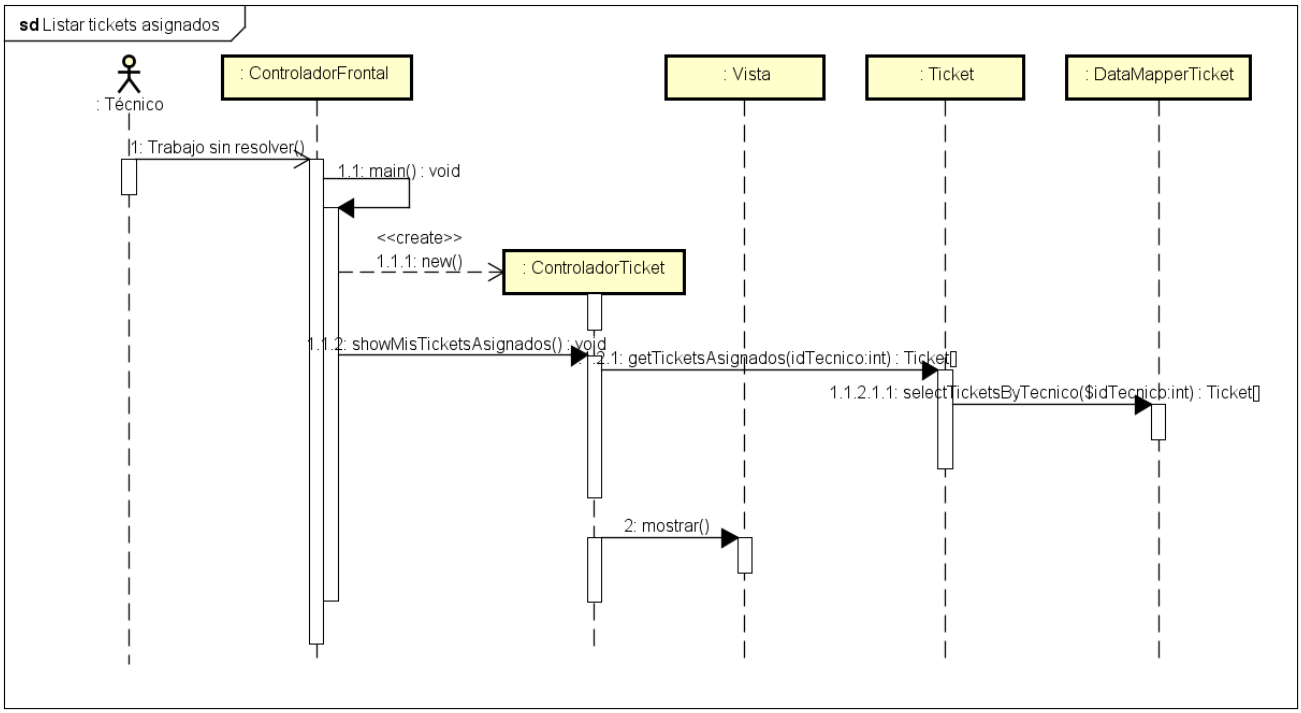


Figura 51: diagrama de secuencia - Listar tickets asignados

5.3.2.11 Diagrama de secuencia: Listar tickets sin asignar

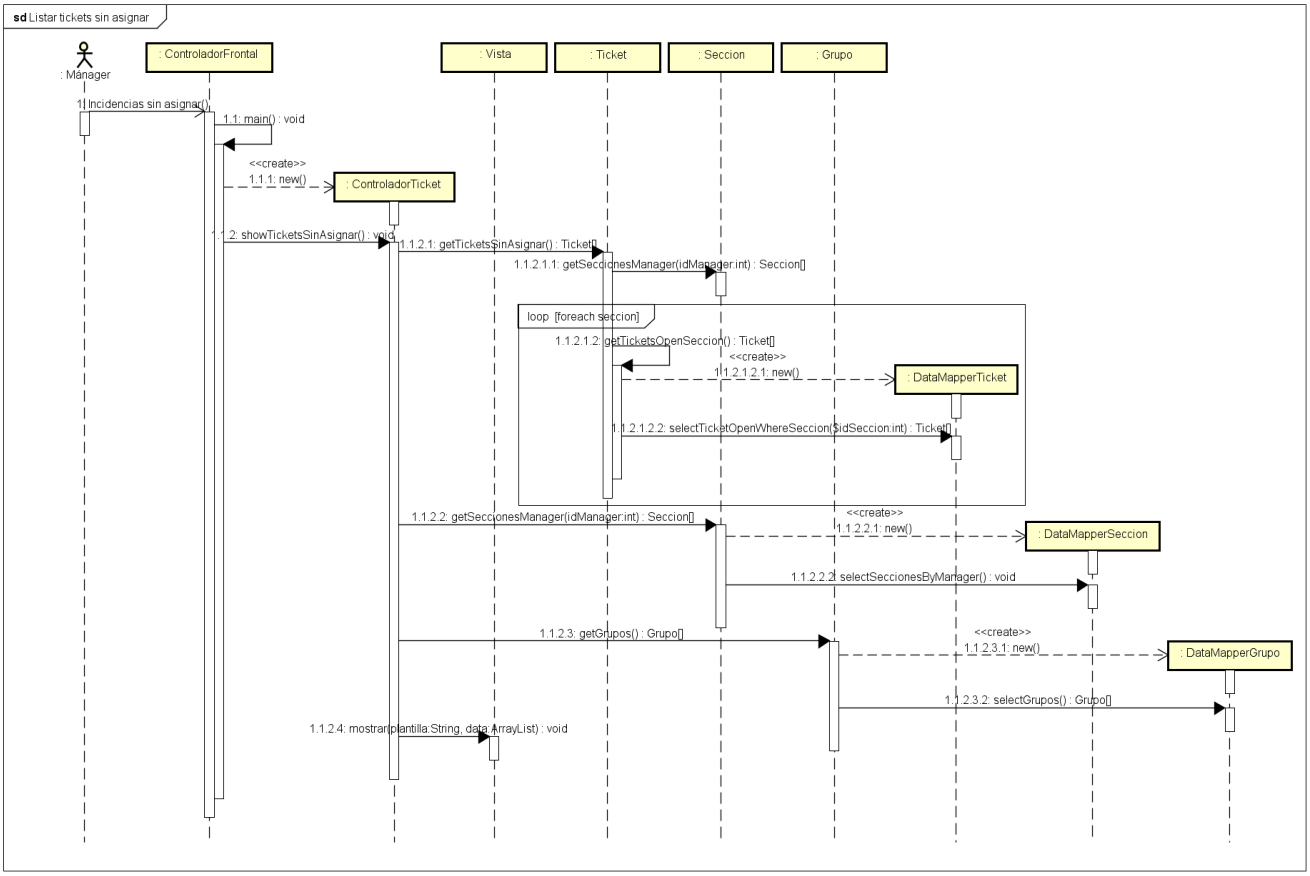


Figura 52: Diagrama de secuencia - Listar tickets sin asignar

5.3.2.12 Diagrama de secuencia: Perfil de usuario

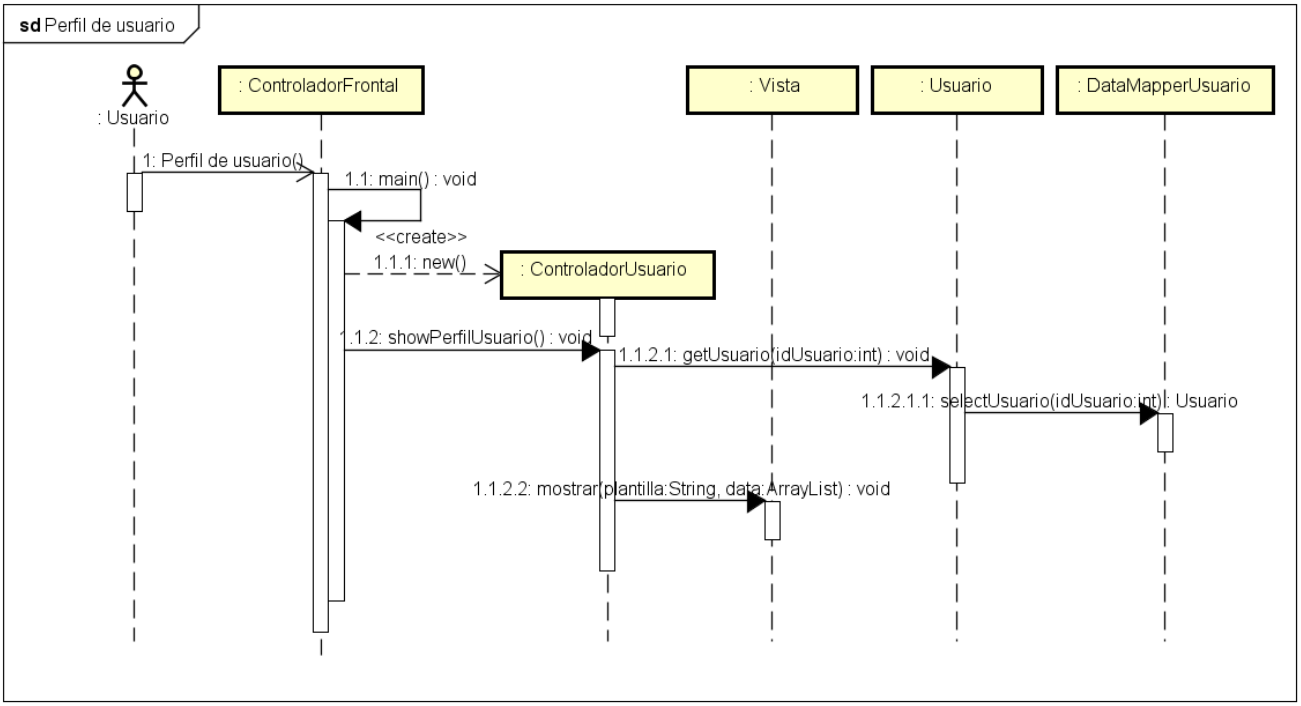


Figura 53: Diagrama de secuencia - Perfil de usuario

6 BLOQUE VI:
IMPLEMENTACIÓN

En esta sección se comentarán los aspectos y decisiones de implementación o desarrollo del software que han influido en el desarrollo del producto, se centra la atención en aquellos más específicos o que se han salido más de lo común, obviando temas más comunes a la hora de implementar en alguna de las tecnologías seleccionadas para el desarrollo de este producto.

6.1 CONTROLADOR FRONTAL

Desde la vista se envían todas las peticiones al Controlador Frontal. Esto se hace por medio del método GET de HTTP, en la misma URL. Por ejemplo, a la hora de solicitar que se muestre el formulario para la creación de un nuevo ticket la petición quedaría: */index.php?controlador=Ticket&accion=formNuevoTicket.* De esta forma se está indicando al Controlador Frontal que el controlador que se necesita utilizar es el controlador de tickets, y la acción que se necesita ejecutar es *formNuevoTicket.*

Desde el controlador frontal se recoge este contenido enviado por la vista (Figura 54). Posteriormente, una vez se tienen el controlador y la acción deseados, se construye la llamada al mismo (Figura 55). Una vez se envían los datos del formulario para que se procesen, se generaría una nueva petición HTTP como se ve a continuación: /index.php?controlador=Ticket&accion=crearTicket. Esta nueva petición será de nuevo procesada por el Controlador Frontal, invocando este de nuevo al controlador y la acción adecuados. Este proceso será el que se sigua para todas las sucesivas peticiones que el usuario genere sucesivamente a lo largo de su uso de la aplicación.

De esta forma se logra tener un punto de recepción centralizado tanto de todas las peticiones HTTP como de las peticiones XMLHTTP (asíncronas - AJAX). La parte de AJAX se verá en la siguiente sección (sección 6.2).

6.2 SOLICITUDES ASÍNCRONAS AL SERVIDOR: AJAX

Algunas funcionalidades incorporan la posibilidad de comunicarse con el servidor sin necesidad de recargar la página. Esto se ha hecho por medio de AJAX: el controlador frontal recibe tanto las peticiones HTTP, como las peticiones XMLHTTP, delegando la responsabilidad en los diferentes controladores.

Se verá un ejemplo de la implementación de esta tecnología. El ejemplo será la carga de datos de un desplegable en función de lo que se seleccione en otro desplegable con anterioridad. El comportamiento es el que sigue: en la Figura 56 se ve como en el desplegable *Centro afectado* aparecen los centros pertenecientes a la sección *Campus Miguel Delibes*, al modificar el contenido del desplegable *Sección afectada*, automáticamente y de forma transparente, sin necesidad de volver a recargar la página, se cargarán los datos de la nueva sección seleccionada en el desplegable *Centro afectado.* En la Figura 57 se puede ver cómo la modificación del primer desplegable ha supuesto una modificación de los datos contenidos en el segundo desplegable.

Esto también puede tener su aspecto negativo, y es que, si la red está sobrecargada, puede resultar algo lenta esta carga del segundo desplegable.

Esto se ha logrado haciendo que el Controlador Frontal que recoge todas las peticiones HTTP sea también capaz de recoger peticiones XMLHTTP, haciendo uso de una función is_ajax (Figura 58), que devolverá true si se trata de una petición XMLHTTP y false en caso contrario.

Si la petición es AJAX, se tomarán los datos enviados por el método POST: acción y controlador, y posteriormente se construirá la llamada al controlador y la acción adecuados tal y como se hacía con las peticiones HTTP (Figura 59).

```
// Formamos el nombre del Controlador o en su defecto, tomamos que es el ControladorCliente
                    if (! empty ( $_GET ['controlador'] ))
                            $controllerName = "Controlador" . $_GET ['controlador'];
                    else
                            $controllerName = "ControladorCliente";
                            // Lo mismo sucede con las acciones, si no hay accion, tomamos paginaVacia
como accion
                    if (! empty ( $_GET ['accion'] ))
                            $actionName = $_GET ['accion'];
                    else
                            $actionName = "showPaginaVacia";
```

Figura 54: Obtención del controlador específico

```
if (is_file ( $controllerPath ))
                    require $controllerPath;
            else
                    // Si no existe la clase que buscamos y su accion, tiramos un error 404
                    die ( 'El controlador no existe - 404 not found' );

                    // Si <controllerName>-><actionName> es una función ejecutable
            if (is_callable ( array (
                            $controllerName,
                            $actionName
            ) ) == false) {
                    trigger_error ( $controllerName . '->' . $actionName . `` no existe',
E_USER_NOTICE );
                    return false;
            }
```

Figura 55: Construcción de la llamada al controlador

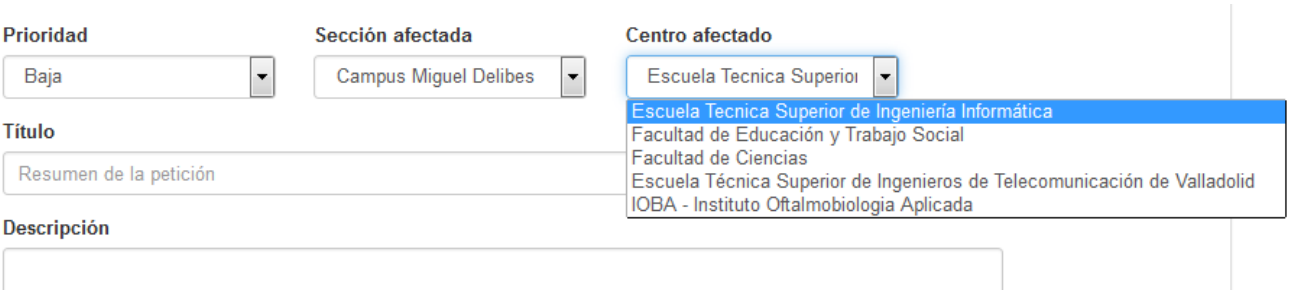


Figura 56: Desplegable para Campus Delibes

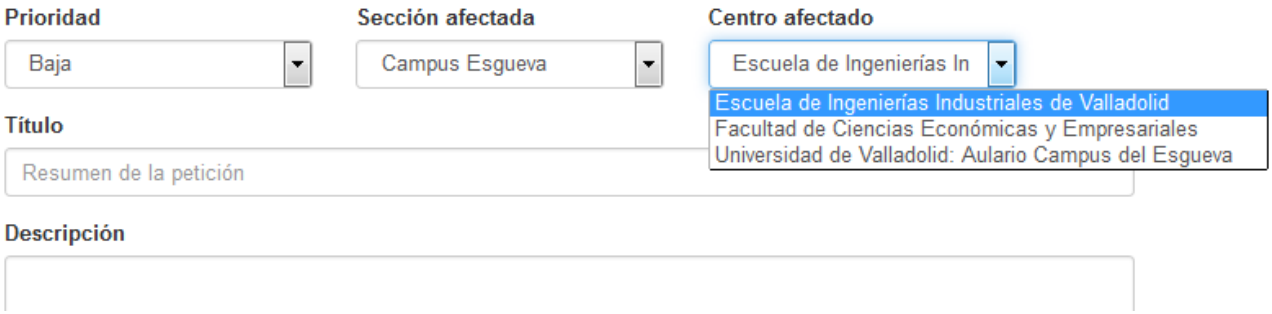


Figura 57: Desplegable para Campus Esgueva

```
function is_ajax() {
                    return isset ( $_SERVER ['HTTP_X_REQUESTED_WITH'] ) && strtolower ( $_SERVER
['HTTP_X_REQUESTED_WITH'] ) == 'xmlhttprequest';
              }
```

Figura 58: Función is_ajax

```
if (is_ajax ()) {

                    if (isset ( $_POST ['accion'] ) && ! empty ( $_POST ['accion'] )) {
                           $actionName = $_POST ['accion'];
                           $controllerName = $_POST ['controlador'];
                    }
```

Figura 59: Construcción de una llamada AJAX

6.3 Interfaz Multiusuario

Dada la necesidad de disponer de una interfaz de usuario diferente en función del rol del usuario que está dentro del sistema actualmente, se ha necesitado disponer de un *layout* o diseño de la interfaz diferente para cada tipo. A modo de disminuir la complejidad de este aspecto se ha optado por definir un área para el contenido de la aplicación y otro área para los menús. El contenido que cambiará con el tipo de usuario será el área de menús, que serán los que permitan al usuario navegar a través de las diferentes funcionalidades.

De esta manera a la hora de hacer login en la aplicación, el sistema detectará de qué tipo es consultando a la base de datos y posteriormente se almacenará en sesión este tipo, mostrándole al usuario el layout que le corresponde en todo momento mientras permanezca con su sesión activa.

6.4 Tratamiento de fechas

El tratamiento de fechas es algo importante a tener en cuenta cuando se trabaja con herramientas de diferente naturaleza como puede ser el lenguaje de postprocesado PHP y un gestor de bases de datos MySQL.

En este caso, las fechas en la base de datos se almacenarán en las tablas con el tipo datetime, este tipo espera la entrada de la fecha en formato “YYYY-mm-dd hh:mm:ss”. Mientras que PHP trabajará con el formato timestamp para una mayor precisión y facilidad de manejo interna de las fechas, el contar con la fecha en este formato de forma interna en PHP facilita la aplicación de un gran número de funciones interesantes con las fechas. Por tanto, cada vez que se obtengan datos de la base de datos habrá que transformarlos a timestamp, y cada vez que se introduzcan fechas en la base de datos, habré que transformar la fecha de timestamp al formato que la base de datos espera.

Para ilustrar este comportamiento se tomará como ejemplo el caso de los tickets, se ha implementado el método de la clase Ticket getFecha() (Figura 60). Si no se pasa ningún parámetro, devolverá la fecha en formato timestamp, si se le pasa un parámetro, se devolverá la fecha formateada de acuerdo con la cadena pasada por dicho parámetro.

Para hacer la operación contraria, es decir, pasar desde el formato de la base de datos al formato timestamp, basta con hacer uso de una función de conversión, en este caso strtotime() de PHP (Figura 61).

6.5 TRATAMIENTO DE CARACTERES

Al igual que las fechas, el tratamiento de caracteres suele dar problemas al usar herramientas de diferente naturaleza. En este trabajo ha sido necesario sincronizar el conjunto de caracteres tanto en la herramienta de desarrollo Eclipse, como en la base de datos y las vistas de las diferentes páginas para su adecuada visualización a la hora de disponer de caracteres propios de un determinado idioma (el castellano en este caso).

En cada una de las vistas es necesario que aparezca la siguiente etiqueta meta:

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

Con esta etiqueta se da a entender al intérprete que el conjunto de caracteres codificado pertenece al estándar utf-8, se ha ubicado esta etiqueta en el head de cada layout diferente (los layouts de cliente, técnico y mánager contendrán el resto de vistas en su interior).

Del mismo modo, hay que configurar el entorno de desarrollo para que reconozca este conjunto de caracteres. Para el desarrollo de este trabajo se ha hecho uso de la herramienta Eclipse Mars, un entorno Eclipse con los complementos adecuados para el desarrollo de aplicaciones empleando PHP. Para configurar el conjunto de caracteres en este entorno es necesario hacerlo desde las *propiedades del proyecto* (Figura 62).

Por último, falta configurar el cotejamiento de la base de datos. Este se ha puesto en *utf8_general_ci* para todas las tablas de la base de datos como se ve en la Figura 63.

6.6 ENVÍO DE FICHEROS AL SERVIDOR

En la aplicación se permite el envío de ficheros por medio de FTP al servidor. Estos ficheros son las imágenes que cada usuario puede subir para completar su perfil de usuario con un avatar. Por medio de una etiqueta input de tipo *file* es posible indicar que lo que se está enviando es un fichero de datos (protocolo FTP).

```
<input type="file" name="foto">
```

Una vez en el controlador, se evalúa si la imagen se puede aceptar en el servidor, se rechazarán las imágenes que no sean jpg y que excedan un tamaño definido como una constante global con valor 5000000 bytes (5 MB aproximadamente) con el objetivo de no saturar la capacidad del servidor con estos archivos. Si todo ha ocurrido como se ha esperado, la imagen se guardará en una carpeta del servidor con el identificador de usuario único como nombre, para así poder acceder a ella fácilmente por medio del valor almacenado en sesión del usuario actualmente en ella. En caso de que el usuario ya tenga una imagen guardada previamente, esta se eliminará y se sustituirá por la nueva para evitar los problemas de almacenamiento en el servidor anteriormente comentados

```php
public function getFecha() {
		$totalArgumentos = func_num_args ();
		switch ($totalArgumentos) {
			// Devuelve la fecha en formato EPOCH
			case 0 : {
					$fecha = $this->fecha;
					break; }
				// Devuelve una lista con los centros de una determinada sección
			case 1 : {
					$formato = func_get_arg ( 0 );
					$fecha = date($formato,$this->getFecha());
					break; }
		}
		return $fecha;
	}
```

Figura 60: Función get_fecha

```
Strtotime($row[“fecha”])
```

Figura 61: Función Strtotime()

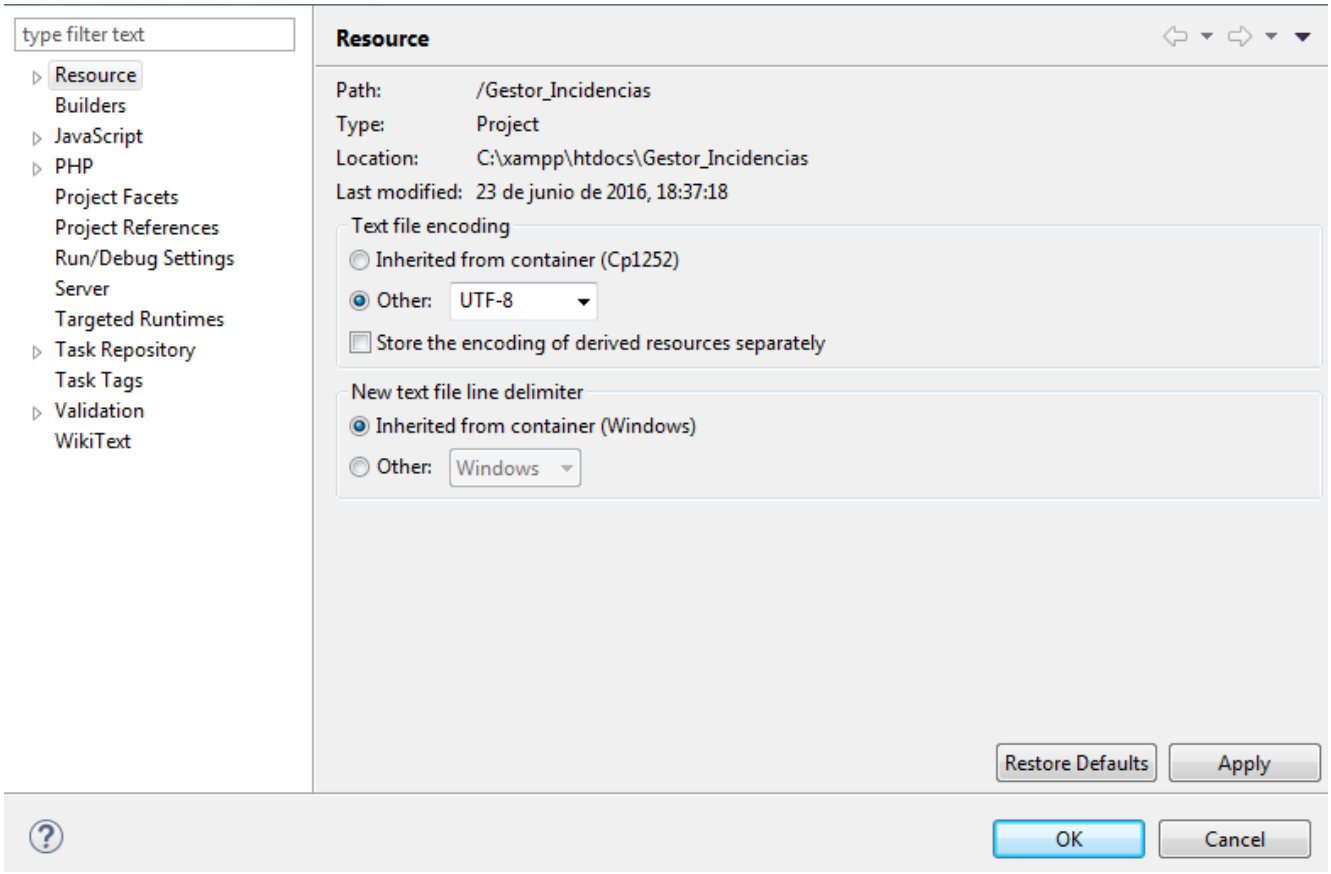


Figura 62: Configuración UTF-8 en Eclipse

Tabla	Acción	Filas	Tipo	Cotejamiento	Tamaño	Residuo a depurar
centro	Examinar Estructura Buscar Insertar Vaciar Eliminar	16	InnoDB	utf8_general_ci	32 KB	-
grupo	Examinar Estructura Buscar Insertar Vaciar Eliminar	12	InnoDB	utf8_general_ci	16 KB	-
modificacion	Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	utf8_general_ci	80 KB	-
seccion	Examinar Estructura Buscar Insertar Vaciar Eliminar	7	InnoDB	utf8_general_ci	48 KB	-
ticket	Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	utf8_general_ci	48 KB	-
usuario	Examinar Estructura Buscar Insertar Vaciar Eliminar	19	InnoDB	utf8_general_ci	80 KB	-
6 tablas	Número de filas	60	InnoDB	utf8_general_ci	304 KB	0 B

Figura 63: Configuración UTF-8 en MySQL

6.7 RESPONSIVENESS

Como se ha comentado para lograr la capacidad de que la página sea *responsive*, se adapte al tamaño y resolución del dispositivo en cuestión, se ha empleado la solución Bootstrap.

Se ha elegido una plantilla que vaya acorde a las características de la interfaz definidas en la parte de requisitos. La que más se ha ajustado a estas necesidades es la plantilla SBAdmin 2, disponible de forma libre y gratuita [28]. Se ha agregado el contenido descargado al espacio de trabajo, el contenido se resume en la Figura 64.

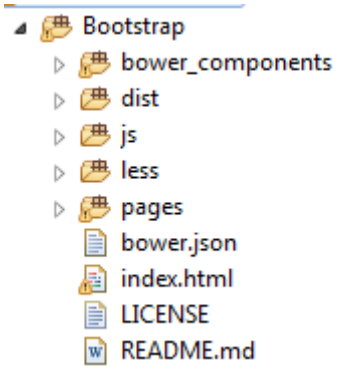


Figura 64: Estructura Bootstrap

Basta con incorporar en los diferentes layouts los siguientes enlaces para los estilos customizados, es decir la plantilla:

```
<link href="../Gestor_Incidencias/Bootstrap/dist/css/sb-admin-2.css" rel="stylesheet" />
```

Para incorporar las librerías de los menús:

```
<link href="../Gestor_Incidencias/Bootstrap/bower_components/metisMenu/dist/metisMenu.min.css"
      rel="stylesheet" />
```

Para incorporar las librerías de Bootstrap:

```
<link href="../Gestor_Incidencias/Bootstrap/bower_components/bootstrap/dist/css/bootstrap.min.css"
      rel="stylesheet" />
```

Y para las páginas que utilicen tablas (DataTable):

```
<link       href="./Bootstrap/bower_components/datatables-responsive/css/responsive.dataTables.scss"
rel="stylesheet">
<link                                                href="./Bootstrap/bower_components/datatables-
plugins/integration/bootstrap/3/dataTables.bootstrap.css" rel="stylesheet">
```

```
$("#tipo").change(function(){
		updateInputs();
	});
```

Figura 65: método updateInputs()

```
function updateInputs(output) {
	if ($("#tipo").val() == "Cliente"){
		$("#inputGroupSupervisor").hide("slow");
		$("#inputGroupCentro").show("slow");
		$("#inputGroupGrupo").hide("slow");
	}
	if ($("#tipo").val() == "Tecnico"){
		$("#inputGroupSupervisor").show("slow");
		$("#inputGroupCentro").hide("slow");
		$("#inputGroupGrupo").show("slow");
	}
	if ($("#tipo").val() == "Manager"){
		$("#inputGroupSupervisor").hide("slow");
		$("#inputGroupCentro").hide("slow");
		$("#inputGroupGrupo").hide("slow");
	}
}
```

Figura 66: Dinamismo de los campos del formulario

6.8 LIBRERÍAS AVANZADAS

Se ha hecho uso de librerías de JavaScript, especialmente de jQuery, para lograr efectos en la presentación de la aplicación de una forma sencilla y eficaz. A modo de ejemplo se detallará cómo se ha implementado el formulario de creación de usuario, que cambiará tanto el aspecto como los datos solicitados en función del tipo de usuario que se quiera crear. Por ejemplo, si se quiere crear un cliente se solicitará el centro al que pertenece, en cambio, si se desea crear un técnico, habrá que introducir su mánager responsable, así como el grupo de técnicos al que pertenece. En la Figura 67 se ve cómo queda el formulario si el nuevo usuario que se quiere crear es de tipo *cliente*, en la Figura 68 se muestra el formulario para la introducción de un usuario de tipo *técnico*, y en la Figura 69 se ve el formulario para la introducción de un usuario de tipo *mánager*.

Esta funcionalidad se ha logrado de la siguiente forma: Cada vez que un elemento con el ID = “tipo” (desplegable de tipos) cambie de valor, se ejecutará la función de jQuery updateInputs() (Figura 65), que redibujará los campos que aparecen en el formulario.

Figura 67: Introducción de cliente

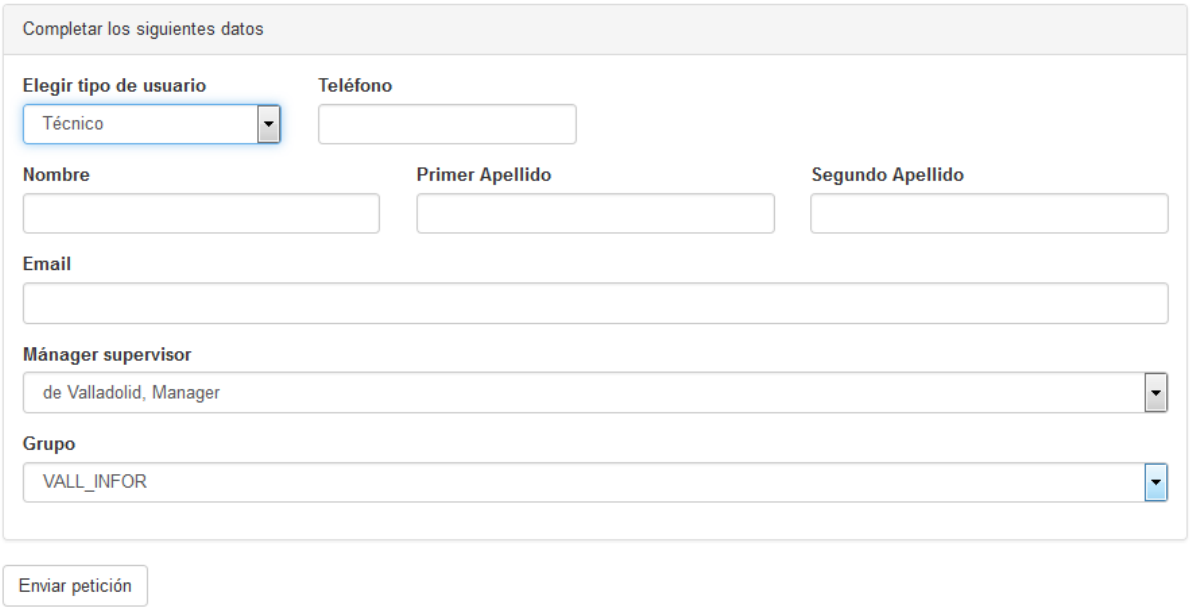


Figura 68: Introducción de técnico

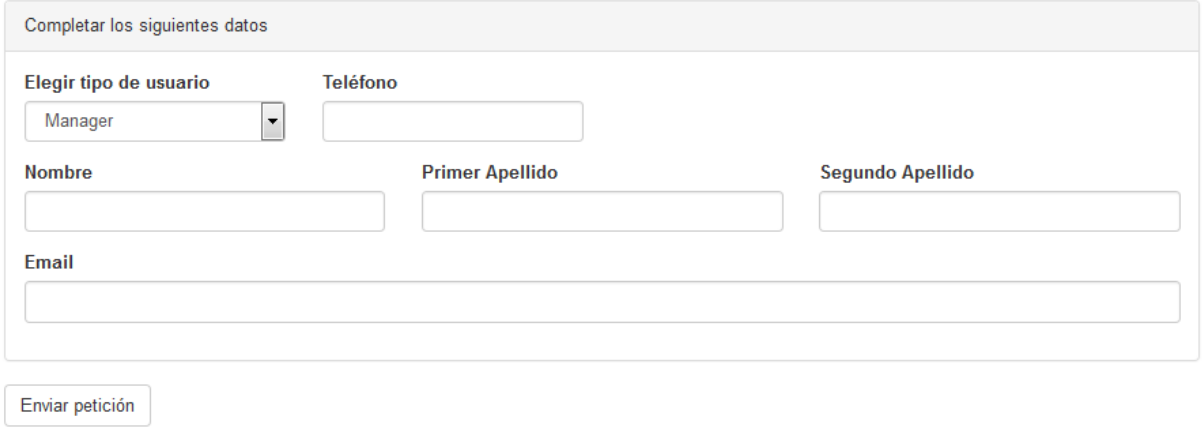


Figura 69:Introducción de manager

6.9 CONEXIÓN SEGURA

Al ser una aplicación que maneja contraseñas de los usuarios es imprescindible establecer un nivel de seguridad en la conexión para así evitar que un usuario malintencionado pueda obtener dichas contraseñas por medio de un ataque del tipo *man in the middle* o ataque de intermediario. Este tipo de ataques consisten en que una entidad, por ejemplo, un programa *sniffer*, intercepta la información que se transmite entre el cliente y el servidor. Esta información debe ir cifrada por medio de un canal seguro para evitar que si se intercepta la información sea difícilmente descifrable. Para establecer una conexión segura por medio del protocolo SSL sobre la capa HTTP son necesarios dos pasos: contar con un certificado SSL y configurar el servidor web para que las páginas protegidas con contraseña sean accesibles por medio de HTTPS.

- **Creación del certificado SSL y la contraseña privada del servidor**

Se hará uso de herramientas incorporadas con XAMPP para facilitar esta tarea. XAMPP proporciona un fichero batch que ayuda a crear el nuevo certificado con claves de encriptación aleatorias. Desde una ventana de consola cmd:

```
cd C:\xampp\apache
makecert
```

Se preguntará por una clave y una serie de datos que compondrán el certificado en su totalidad. De todos ellos cabe destacar el *Common Name*, en el cual hay que indicar la IP o el DNS del servidor en el que está alojado el sitio web, en este caso será *localhost*. Una vez completado el proceso el script *makecert.bat* moverá los certificados y la clave privada del servidor a los directorios adecuados.

Para evitar que en el navegador aparezca una alerta cada vez que se intente acceder es recomendable importar el certificado ubicado en c:\xampp\apache\conf\ssl.crt\server.crt en dicho navegador.

- **Editar la configuración de Apache**

Basta con editar el fichero de configuración *httpd-xampp.conf* y añadir la línea *SSLRequiredSSL* en cada uno de los directorios del servidor web que se quieren proteger:

```
<Directory "C:/xampp/htdocs/gestor_Incidencias/">
    <IfModule php5_module>
       <Files "status.php">
              php_admin_flag safe_mode off
       </Files>
    </IfModule>
    AllowOverride AuthConfig
    SSLRequireSSL
</Directory>
```

Por último, es recomendable redirigir las peticiones HTTP a peticiones HTTPS para que al tratar de acceder por medio de http se redirija automáticamente a https en lugar de obtener un error de servidor ya que la página no se encontrará. Para ello es necesario tener habilitado el módulo *mod_rewrite* en el archivo de configuración *httpd.conf* de Apache:

```
LoadModule rewrite_module modules/mod_rewrite.so
```

Y copiar las siguientes líneas en el fichero *httpd-xampp.conf*:

```
<IfModule mod_rewrite.c>
    RewriteEngine On

    # Redirect /gestor_incidencias folder to https
    RewriteCond %{HTTPS} !=on
    RewriteCond %{REQUEST_URI} gestor_incidencias
    RewriteRule ^(.*) https://%{SERVER_NAME}$1 [R,L]

    # Redirect /phpMyAdmin folder to https
    RewriteCond %{HTTPS} !=on
    RewriteCond %{REQUEST_URI} phpmyadmin
    RewriteRule ^(.*) https://%{SERVER_NAME}$1 [R,L]

    # Redirect /security folder to https
    RewriteCond %{HTTPS} !=on
    RewriteCond %{REQUEST_URI} security
    RewriteRule ^(.*) https://%{SERVER_NAME}$1 [R,L]

    # Redirect /webalizer folder to https
    RewriteCond %{HTTPS} !=on
    RewriteCond %{REQUEST_URI} webalizer
    RewriteRule ^(.*) https://%{SERVER_NAME}$1 [R,L]
</IfModule>
```

7 BLOQUE VII:
PRUEBAS Y RESULTADOS

Para realizar las pruebas previamente se ha cargado la base de datos con una cantidad de datos suficiente para probar todas las funcionalidades de la herramienta adecuadamente. Se han añadido una serie de usuarios y tickets ficticios, junto con un conjunto de secciones y centros inspirados en la Universidad de Valladolid. Las secciones corresponderán a los diferentes campus, y los centros a cada uno de los centros localizados dentro de estos campus. Por ejemplo, una de las secciones corresponderá al Campus Miguel Delibes, y los centros corresponderán a las facultades, centros de investigación y bibliotecas de los que este campus consta.

Se seguirá un plan de pruebas basado en dos partes: pruebas unitarias (pruebas de caja negra) y pruebas guiadas por casos de prueba (pruebas de sistema). Por último, se hará un resumen de las características funcionales haciendo uso de las métricas definidas en la sección 4.1.

7.1 PRUEBAS UNITARIAS

En este tipo de pruebas se comprobará el funcionamiento del sistema "pieza a pieza", es decir, cada una de las funcionalidades unitarias o atómicas de las que consta la aplicación con la obtención de una serie de clases de equivalencia en función de las entradas posibles que el usuario pueda introducir y se analizarán los resultados en función de las salidas proporcionadas por el sistema. A partir de estas clases de equivalencia se elaborará un plan de pruebas de caja negra para cada uno de los datos introducidos por el usuario en la aplicación.

7.1.1 Clases de equivalencia

Se definirán en primer lugar unas clases de equivalencia válidas para cada uno de los inputs que el usuario introduce en la aplicación en forma de texto alfanumérico, acotándose así los valores que se pueden introducir en cada uno de los campos, y qué campos se tomarán como válidos. Para estas clases de equivalencia se tomarán los datos en los que el usuario introduce datos de forma libre, por tanto, no se tendrán en cuenta los campos que vienen ya cerrados y formateados por parte de la aplicación, como por ejemplo los desplegables con las secciones y los centros.

Estas clases de equivalencia ayudarán a identificar los datos frontera, que son los más susceptibles a dar condiciones de error o fallo. Estos datos frontera serán los valores máximos y mínimos que aceptará como entrada la aplicación.

Email del usuario:

Condiciones de entrada	Clases de equivalencia válidas	Clases de equivalencia no válidas
Comprendido entre 1 y 50 caracteres (n)	$1 \leq n \leq 50$	$n < 1$
		$n > 50$
Formato de email obligatorio	usuario@dominio.xxx	usuario
		usuario@
		usuario@dominio
		@dominio.xxx

Tabla 19: Clases de equivalencia: email de usuario

Password del usuario:

Condiciones de entrada	Clases de equivalencia válidas	Clases de equivalencia no válidas
Texto comprendido entre 1 y 30 caracteres (n)	$1 \leq n \leq 30$	$n < 1$
		$n > 30$

Tabla 20: Clases de equivalencia: Password

Primer Apellido del usuario:

Condiciones de entrada	Clases de equivalencia válidas	Clases de equivalencia no válidas
Texto comprendido entre 1 y 20 caracteres (n)	1 ≤ n ≤ 20	n < 1
		n > 20

Tabla 21: Clases de equivalencia: Primer apellido

Segundo Apellido del usuario:

Condiciones de entrada	Clases de equivalencia válidas	Clases de equivalencia no válidas
Texto comprendido entre 1 y 20 caracteres (n)	1 ≤ n ≤ 20	n < 1
		n > 20

Tabla 22: Clases de equivalencia: Segundo apellido

Teléfono del usuario:

Condiciones de entrada	Clases de equivalencia válidas	Clases de equivalencia no válidas
Texto comprendido entre 1 y 10 caracteres (n)	1 ≤ n ≤ 10	n < 1
		n > 10
Caracteres numéricos obligatorios	Dato numérico	Dato no numérico

Tabla 23: Clases de equivalencia: teléfono

Título del ticket:

Condiciones de entrada	Clases de equivalencia válidas	Clases de equivalencia no válidas
Texto comprendido Entre 1 y 200 caracteres (n)	1 ≤ n ≤ 200	n < 1
		n > 200

Tabla 24: Clases de equivalencia: Título del ticket

Descripción del ticket:

Condiciones de entrada	Clases de equivalencia válidas	Clases de equivalencia no válidas
Texto obligatorio sin límite	1 ≤ n	n < 1

Tabla 25: Clases de equivalencia: Descripción del ticket

7.1.2 Pruebas

Se hará uso de los requisitos funcionales identificados en la sección IV para diseñar el plan de pruebas. Para cada una de las pruebas se definirán los siguientes aspectos de la misma:

- Nombre
- Requisito funcional relacionado
- Datos de entrada
- Datos de salida esperados
- Datos de salida obtenidos
- Comentario

Las pruebas se han realizado bajo las siguientes condiciones:

- Sistema operativo Windows 7.
- Navegador Web: Mozilla Firefox 48.0.2

7.1.2.1 Índice de la batería de pruebas

1. Área de Cliente
   1.1. Login
   1.2. Creación de un ticket
   1.3. Listado de tickets a su nombre
   1.4. Cambiar contraseña
   1.5. Cerrar sesión
   1.6. Ver perfil
   1.7. Cambiar foto
   1.8. Ver detalles de un ticket
2. Área de Técnico
   2.1. Login
   2.2. Creación de un ticket
   2.3. Listado de tickets a su nombre
   2.4. Cambiar contraseña
   2.5. Cerrar sesión
   2.6. Ver perfil
   2.7. Cambiar foto
   2.8. Ver detalles de un ticket
   2.9. Listado de tickets asignados
   2.10. Cambiar ticket asignado a resuelto
3. Área de Mánager
   3.1. Login
   3.2. Creación de un ticket
   3.3. Listado de tickets a su nombre
   3.4. Cambiar contraseña
   3.5. Cerrar sesión
   3.6. Ver perfil
   3.7. Cambiar foto
   3.8. Ver detalles de un ticket
   3.9. Asignar técnico a un ticket
   3.10. Listado de tickets de su sección
   3.11. Creación de usuario
   3.12. Creación de centro
   3.13. Listado de secciones
4. Sistema
   4.1. Cerrar sesión pasados 600 segundos

En total las funcionalidades a probar serán 32.

7.1.2.2 Batería de pruebas

Referencia	P-1.1
Nombre	Login
Requisito mínimo relacionado	RF-01
Datos de entrada	• Email: cliente@cliente.com • Password: cliente
Datos de salida esperados	Se muestra la pantalla de cliente con un mensaje de bienvenida
Datos de salida obtenidos	Pantalla con el menú de cliente y un mensaje de bienvenida
Comentario	Ninguno

Tabla 26: Prueba P-1.1

Referencia	P-1.2
Nombre	Creación de un ticket
Requisito mínimo relacionado	RF-02
Datos de entrada	• Prioridad: Baja • Sección afectada: Campus Miguel Delibes • Centro afectado: Escuela Técnica Superior de Ingeniería Informática • Título: Título de prueba • Descripción: Descripción de prueba: Lorem ipsum dolor sit amet...
Datos de salida esperados	Se crea un ticket en la base de datos con los datos introducidos, se muestra un mensaje de confirmación al usuario.
Datos de salida obtenidos	Mensaje de confirmación del éxito en la transacción a la base de datos mostrado. Registro cargado en la base de datos.
Comentario	Se validan las clases de equivalencia definidas para título y descripción antes del envío del formulario.

Tabla 27: Prueba P-1.2

Referencia	P-1.3
Nombre	Listado de tickets a su nombre
Requisito mínimo relacionado	RF-03
Datos de entrada	• Click en “Mis incidencias”
Datos de salida esperados	Listado con los tickets abiertos a nombre del usuario actualmente en sesión, aportando los datos: número de ticket, título, fecha de apertura, estado.
Datos de salida obtenidos	Tabla clickable y con opción de búsquedas y ordenamientos con los datos esperados.
Comentario	Al hacer click en alguna de las filas, se mostrarán detalles del ticket en cuestión.

Tabla 28: Prueba P-1.3

Referencia	P-1.4
Nombre	Cambiar contraseña
Requisito mínimo relacionado	RF-05
Datos de entrada	• Nueva contraseña: cliente2 • Confirmar nueva contraseña: cliente2
Datos de salida esperados	La contraseña se modifica en la base de datos y se muestra un mensaje de confirmación.
Datos de salida obtenidos	Mensaje de confirmación del éxito en la transacción a la base de datos mostrado. Registro modificado en la base de datos.
Comentario	Ninguno.

Tabla 29: Prueba P-1.4

Referencia	P-1.5
Nombre	Cerrar sesión
Requisito mínimo relacionado	RF-08
Datos de entrada	• Click en “Cerrar sesión”
Datos de salida esperados	Se eliminará la sesión y aparecerá la pantalla de login.
Datos de salida obtenidos	Sesión eliminada y aparece la pantalla de login
Comentario	Ninguno.

Tabla 30: Prueba P-1.5

Referencia	P-1.6
Nombre	Ver perfil
Requisito mínimo relacionado	RF-09
Datos de entrada	• Click en “Perfil de usuario”
Datos de salida esperados	Se muestran los datos del usuario, junto con su foto y la opción de cambiarla.
Datos de salida obtenidos	Todos los datos del usuario se muestran excepto la contraseña.
Comentario	Ninguno.

Tabla 31: Prueba P-1.6

Referencia	P-1.7
Nombre	Cambiar foto
Requisito mínimo relacionado	RF-10
Datos de entrada	• Click en “Perfil de usuario” • Imagen: Koala.jpg (762 KB)
Datos de salida esperados	Se sube la imagen al servidor con el ID del usuario como nombre, y se modifica el registro de la base de datos.
Datos de salida obtenidos	Foto subida correctamente, registro de la base de datos modificado.
Comentario	Ninguno.

Tabla 32: Prueba P-1.7

Referencia	P-1.8
Nombre	Ver detalles de un ticket
Requisito mínimo relacionado	RF-04
Datos de entrada	• Click en alguna fila del listado de tickets
Datos de salida esperados	Se muestran los detalles del ticket: título y descripción.
Datos de salida obtenidos	La salida es la esperada.
Comentario	Ninguno.

Tabla 33: Prueba P-1.8

Referencia	P-2.1
Nombre	Login
Requisito mínimo relacionado	RF-11
Datos de entrada	• Email: tecnicoVALLINFOR@tecnico.com • Password: tecnico
Datos de salida esperados	Se muestra la pantalla de técnico con un mensaje de bienvenida
Datos de salida obtenidos	Pantalla con el menú de técnico y un mensaje de bienvenida
Comentario	Ninguno

Tabla 34: Prueba P-2.1

Referencia	P-2.2
Nombre	Creación de un ticket
Requisito mínimo relacionado	RF-12
Datos de entrada	• Prioridad: Baja • Sección afectada: Campus Miguel Delibes • Centro afectado: Escuela Técnica Superior de Ingeniería Informática • Título: Título de prueba • Descripción: Descripción de prueba: Lorem ipsum dolor sit amet...
Datos de salida esperados	Se crea un ticket en la base de datos con los datos introducidos, se muestra un mensaje de confirmación al usuario.
Datos de salida obtenidos	Mensaje de confirmación del éxito en la transacción a la base de datos mostrado. Registro cargado en la base de datos.
Comentario	Se validan las clases de equivalencia definidas para título y descripción antes del envío del formulario.

Tabla 35: Prueba P-2.2

Referencia	P-2.3
Nombre	Listado de tickets a su nombre
Requisito mínimo relacionado	RF-14
Datos de entrada	• Click en “Mis incidencias”
Datos de salida esperados	Listado con los tickets abiertos a nombre del usuario actualmente en sesión, aportando los datos: número de ticket, título, fecha de apertura, estado.
Datos de salida obtenidos	Tabla clickable y con opción de búsquedas y ordenamientos con los datos esperados.
Comentario	Al hacer click en alguna de las filas, se mostrarán detalles del ticket en cuestión.

Tabla 36: Prueba P-2.3

Referencia	P-2.4
Nombre	Cambiar contraseña
Requisito mínimo relacionado	RF-15
Datos de entrada	• Nueva contraseña: tecnico2 • Confirmar nueva contraseña: tecnico2
Datos de salida esperados	La contraseña se modifica en la base de datos y se muestra un mensaje de confirmación.
Datos de salida obtenidos	Mensaje de confirmación del éxito en la transacción a la base de datos mostrado. Registro modificado en la base de datos.
Comentario	Ninguno.

Tabla 37: Prueba P-2.4

Referencia	P-2.5
Nombre	Cerrar sesión
Requisito mínimo relacionado	RF-08
Datos de entrada	• Click en “Cerrar sesión”
Datos de salida esperados	Se eliminará la sesión y aparecerá la pantalla de login.
Datos de salida obtenidos	Sesión eliminada y aparece la pantalla de login
Comentario	Ninguno.

Tabla 38: Prueba P-2.5

Referencia	P-2.6
Nombre	Ver perfil
Requisito mínimo relacionado	RF-09
Datos de entrada	• Click en “Perfil de usuario”
Datos de salida esperados	Se muestran los datos del usuario, junto con su foto y la opción de cambiarla.
Datos de salida obtenidos	Todos los datos del usuario se muestran excepto la contraseña.
Comentario	Ninguno.

Tabla 39: Prueba P-2.6

Referencia	P-2.7
Nombre	Cambiar foto
Requisito mínimo relacionado	RF-10
Datos de entrada	• Click en “Perfil de usuario” • Imagen: Koala.jpg (762 KB)
Datos de salida esperados	Se sube la imagen al servidor con el ID del usuario como nombre, y se modifica el registro de la base de datos.
Datos de salida obtenidos	Foto subida correctamente, registro de la base de datos modificado.
Comentario	Ninguno.

Tabla 40: Prueba P-2.7

Referencia	P-2.8
Nombre	Ver detalles de un ticket
Requisito mínimo relacionado	RF-04
Datos de entrada	• Click en alguna fila del listado de tickets
Datos de salida esperados	Se muestran los detalles del ticket, el técnico asignado, el centro y sección afectados.
Datos de salida obtenidos	La salida es la esperada.
Comentario	Ninguno.

Tabla 41: Prueba P-2.8

Referencia	P-2.9
Nombre	Listado de tickets asignados
Requisito mínimo relacionado	RF-14
Datos de entrada	• Click en “Trabajo por resolver”
Datos de salida esperados	Listado con los tickets asignados a nombre del usuario actualmente en sesión, aportando los datos: número de ticket, título, fecha de apertura, estado.
Datos de salida obtenidos	Tabla clickable y con opción de búsquedas y ordenamientos con los datos esperados.
Comentario	Al hacer click en alguna de las filas, se mostrarán detalles del ticket en cuestión.

Tabla 42: Prueba P-2.9

Referencia	P-2.10
Nombre	Cambiar ticket de asignado a resuelto
Requisito mínimo relacionado	RF-14
Datos de entrada	• Acceder a los detalles de un ticket • Solución: Solución de prueba • Comentario: Comentario para solución de prueba
Datos de salida esperados	Registros de la base de datos actualizados con el nuevo de estado del ticket y la nueva modificación
Datos de salida obtenidos	La salida es la esperada
Comentario	Ninguno.

Tabla 43: Prueba P-2.10

Referencia	P-3.1
Nombre	Login
Requisito mínimo relacionado	RF-11
Datos de entrada	• Email: managerVA@manager.com • Password: manager
Datos de salida esperados	Se muestra la pantalla de mánager con un mensaje de bienvenida
Datos de salida obtenidos	Pantalla con el menú de mánager y un mensaje de bienvenida
Comentario	Ninguno

Tabla 44: Prueba P-3.1

Referencia	P-3.2
Nombre	Creación de un ticket
Requisito mínimo relacionado	RF-12
Datos de entrada	• Prioridad: Baja • Sección afectada: Campus Miguel Delibes • Centro afectado: Escuela Técnica Superior de Ingeniería Informática • Título: Título de prueba • Descripción: Descripción de prueba: Lorem ipsum dolor sit amet...
Datos de salida esperados	Se crea un ticket en la base de datos con los datos introducidos, se muestra un mensaje de confirmación al usuario.
Datos de salida obtenidos	Mensaje de confirmación del éxito en la transacción a la base de datos mostrado. Registro cargado en la base de datos.
Comentario	Se validan las clases de equivalencia definidas para título y descripción antes del envío del formulario.

Tabla 45: Prueba P-3.2

Referencia	P-3.3
Nombre	Listado de tickets a su nombre
Requisito mínimo relacionado	RF-14
Datos de entrada	• Click en “Mis incidencias”
Datos de salida esperados	Listado con los tickets abiertos a nombre del usuario actualmente en sesión, aportando los datos: número de ticket, título, fecha de apertura, estado.
Datos de salida obtenidos	Tabla clickable y con opción de búsquedas y ordenamientos con los datos esperados.
Comentario	Al hacer click en alguna de las filas, se mostrarán detalles del ticket en cuestión.

Tabla 46: Prueba P-3.3

Referencia	P-3.4
Nombre	Cambiar contraseña
Requisito mínimo relacionado	RF-15
Datos de entrada	• Nueva contraseña: manager2 • Confirmar nueva contraseña: manager2
Datos de salida esperados	La contraseña se modifica en la base de datos y se muestra un mensaje de confirmación.
Datos de salida obtenidos	Mensaje de confirmación del éxito en la transacción a la base de datos mostrado. Registro modificado en la base de datos.
Comentario	Ninguno.

Tabla 47: Prueba P-3.4

Referencia	P-3.5
Nombre	Cerrar sesión
Requisito mínimo relacionado	RF-08
Datos de entrada	• Click en “Cerrar sesión”
Datos de salida esperados	Se eliminará la sesión y aparecerá la pantalla de login.
Datos de salida obtenidos	Sesión eliminada y aparece la pantalla de login
Comentario	Ninguno.

Tabla 48: Prueba P-3.5

Referencia	P-3.6
Nombre	Ver perfil
Requisito mínimo relacionado	RF-09
Datos de entrada	• Click en “Perfil de usuario”
Datos de salida esperados	Se muestran los datos del usuario, junto con su foto y la opción de cambiarla.
Datos de salida obtenidos	Todos los datos del usuario se muestran excepto la contraseña.
Comentario	Ninguno.

Tabla 49: Prueba P-3.6

Referencia	P-3.7
Nombre	Cambiar foto
Requisito mínimo relacionado	RF-10
Datos de entrada	• Click en “Perfil de usuario” • Imagen: Koala.jpg (762 KB)
Datos de salida esperados	Se sube la imagen al servidor con el ID del usuario como nombre, y se modifica el registro de la base de datos.
Datos de salida obtenidos	Foto subida correctamente, registro de la base de datos modificado.
Comentario	Ninguno.

Tabla 50: Prueba P-3.7

Referencia	P-3.8
Nombre	Ver detalles de un ticket
Requisito mínimo relacionado	RF-04
Datos de entrada	• Click en alguna fila del listado de tickets
Datos de salida esperados	Se muestran los detalles del ticket, el técnico asignado, el centro y sección afectados y la posibilidad de asignárselo a algún técnico.
Datos de salida obtenidos	La salida es la esperada.
Comentario	Ninguno.

Tabla 51: Prueba P-3.8

Referencia	P-3.9
Nombre	Asignar técnico a un ticket
Requisito mínimo relacionado	RF-24
Datos de entrada	• Click en alguna fila del listado de tickets
Datos de salida esperados	Se muestran los detalles del ticket, el técnico asignado, el centro y sección afectados y la posibilidad de asignárselo a algún técnico.
Datos de salida obtenidos	La salida es la esperada.
Comentario	Ninguno.

Tabla 52: Prueba P-3.9

Referencia	P-3.10
Nombre	Listado de tickets de sus secciones
Requisito mínimo relacionado	RF-22
Datos de entrada	• Click en “Incidencias sin asignar”
Datos de salida esperados	Listado con los tickets abiertos y que en ese momento no tienen a ningún técnico asignado aún, aportando los datos: número de ticket, título, sección, fecha de apertura, grupo y técnico asignado.
Datos de salida obtenidos	La salida es la esperada.
Comentario	Ninguno.

Tabla 53: Prueba P-3.10

Referencia	P-3.11
Nombre	Creación de usuario
Requisito mínimo relacionado	RF-28
Datos de entrada	• Click en “Añadir nuevo usuario” • Tipo de usuario: Cliente • Teléfono: 123456789 • Nombre: Marcos • Primer apellido: García • Segundo apellido: Alonso • Email: marcos.garcia.alonso@outlook.es • Sección a la que pertenece: Campus Miguel Delibes • Centro al que pertenece: Escuela de Ingeniería Informática
Datos de salida esperados	Se crea un usuario en la base de datos con los datos introducidos, se muestra un mensaje de confirmación al usuario y se le envía un correo electrónico con la contraseña.
Datos de salida obtenidos	La salida es la esperada.
Comentario	Ninguno.

Tabla 54: Prueba P-3.11

Referencia	P-3.12
Nombre	Creación de centro
Requisito mínimo relacionado	RF-28
Datos de entrada	• Click en “Añadir centro” • Sección: Campus Miguel Delibes • Teléfono: 983423559 • Nombre: IOBA Instituto de Oftalmobiología Aplicada • Dirección: Paseo de Belén, 17, 47011 Valladolid
Datos de salida esperados	Se crea un centro en la base de datos con los datos introducidos, se muestra un mensaje de confirmación al usuario.
Datos de salida obtenidos	La salida es la esperada.
Comentario	Ninguno.

Tabla 55: Prueba P-3.12

Referencia	P-3.13
Nombre	Listado de secciones
Requisito mínimo relacionado	RF-28
Datos de entrada	• Click en “Secciones”
Datos de salida esperados	Listado con las secciones bajo el control del mánager actualmente en sesión, junto con los centros de los que consta dicha sección.
Datos de salida obtenidos	La salida es la esperada.
Comentario	Ninguno.

Tabla 56: Prueba P-3.13

7.2 PRUEBAS DE SISTEMA

La función de estas pruebas será la de comprobar que se cumplen los requisitos definidos en la sección IV del documento. Para ello se hará uso de los casos de uso identificados, que surgen de esos requisitos. Se seguirá el flujo definido para el caso ideal para cada uno de los casos de uso, esto es, sin entrar en los flujos alternativos.

Caso de uso	Resultado esperado	Resultado obtenido
UC1: Login	El usuario entra en la aplicación con su vista correspondiente	Correcto
UC2: Abrir ticket	Se ha creado un ticket en la base de datos a nombre del creador	Correcto
UC3: Listar tickets a mi nombre	Lista con los tickets abiertos por el usuario activo	Correcto
UC4: Listar tickets asignados	Lista con los tickets asignados al usuario activo (técnico)	Correcto
UC5: Listar tickets de mi sección	Lista con los tickets creados en la sección controlada por el usuario activo (mánager)	Correcto
UC6: Detalles ticket	Detalles del ticket seleccionado	Correcto
UC7: Asignar técnico	El ticket seleccionado se asigna al técnico seleccionado	Correcto
UC8: Crear usuario	Se ha creado un usuario en la base de datos	Correcto
UC9: Perfil de usuario	Detalles del usuario activo	Correcto
UC10: Cambiar contraseña	La contraseña del usuario activo se modifica en la base de datos	Correcto
UC11: Buscar ticket	Detalles del ticket introducido en el campo de búsqueda	Correcto
UC12: Cerrar sesión	Página de login y sesión destruida	Correcto

Tabla 57: Pruebas del sistema

7.3 RESULTADOS DE LAS MÉTRICAS

En esta sección se evaluará si los resultados de las características funcionales de la aplicación son válidos en función de las métricas definidas con anterioridad (ver sección 4.1). En la Tabla 58 se resumen los resultados obtenidos. Todas las medidas han sido obtenidas a partir de la aplicación alojada en la máquina virtual proporcionada por la Universidad de Valladolid, accesible a través de la URL:

http://pasarela.lab.inf.uva.es:20072/Gestor_Incidencias/index.php

En los casos de uso que requieren más de una recarga de página, se ha obtenido la media aritmética de estos tiempos para obtener la medida del tiempo de respuesta.

Todos los resultados obtenidos para el número de clics y de errores han sido buenos en todos los casos de uso estudiados.

Por otro lado, el tiempo de respuesta ha sido malo para el caso de uso abrir ticket, aunque solo ha sido por 4 milésimas de segundo, por lo que no se puede considerar un resultado malo en sí, ya que, en diferentes condiciones en el estado de la red, dependiendo de la hora, la carga de trabajo que tenga, u otras variables, este tiempo puede variar (tanto para mejorar como para empeorar). Por último, los tiempos de respuesta para los casos de uso “Asignar técnico” y “Crear usuario” han sido aceptables.

Como conclusión, destacar que los casos de uso que no han dado resultados buenos en los tiempos de respuesta, han sido los que implican inserciones en la base de datos, a excepción del caso de uso “Cambiar contraseña” (probablemente debido a que se trata de una inserción muy sencilla). Esto es probablemente debido a la forma en que el SGDB trata a las tablas, indizando las consultas más comunes para optimizar los tiempos de respuesta.

Caso de Uso	Métrica	Resultado
UC1: Login	Tiempo de Respuesta (s)	0,073
	Número de clics	1
	Número de errores	0
UC2: Abrir ticket	Tiempo de Respuesta (s)	1,004
	Número de clics	2
	Número de errores	0
UC3: Listar tickets a mi nombre	Tiempo de Respuesta (s)	0,132
	Número de clics	1
	Número de errores	0
UC4: Listar tickets asignados	Tiempo de Respuesta (s)	0,043
	Número de clics	1
	Número de errores	0
UC5: Listar tickets de mi sección	Tiempo de Respuesta (s)	0,050
	Número de clics	2
	Número de errores	0
UC6: Detalles ticket	Tiempo de Respuesta (s)	0,051
	Número de clics	2
	Número de errores	0
UC7: Asignar técnico	Tiempo de Respuesta (s)	0,800
	Número de clics	3
	Número de errores	0
UC8: Crear usuario	Tiempo de Respuesta (s)	0,132
	Número de clics	3
	Número de errores	0
UC9: Perfil usuario	Tiempo de Respuesta (s)	0,041
	Número de clics	2
	Número de errores	0
UC10: Cambiar contraseña	Tiempo de Respuesta (s)	0,063
	Número de clics	3
	Número de errores	0
UC11: Buscar ticket	Tiempo de Respuesta (s)	0,050
	Número de clics	1
	Número de errores	0
UC12: Cerrar sesión	Tiempo de Respuesta (s)	0,037
	Número de clics	2
	Número de errores	0

Tabla 58: Resultados de las métricas

8 BLOQUE VIII:
CONCLUSIONES Y TRABAJO FUTURO

8.1 CONCLUSIONES

En esta parte final del documento se hará una valoración general de toda la parte de planificación, aprendizaje y desarrollo del producto, desde la idea inicial, hasta su implementación.

Existen gran número de aplicaciones de tipo HelpDesk, cada una de ellas con sus particularidades, generalmente girando en torno a la misma idea de centralizar el intercambio de notificaciones de incidencias y sus resoluciones en un mismo punto. Es muy habitual en grandes empresas la utilización de estas aplicaciones como herramienta de trabajo de los diferentes equipos de IT.

En este trabajo se ha tratado de dar una alternativa más flexible y más accesible a personas que no estén familiarizadas con este tipo de aplicaciones, haciéndose un producto más limpio, simplificado y directo que las herramientas tradicionales que se puedan encontrar en el mercado. Para que de este modo se pueda introducir en ámbitos menos especializados que el puramente empresarial tales como instituciones públicas (colegios, universidades), asociaciones u organizaciones pequeñas.

- Se ha logrado una aplicación tipo HelpDesk a la que se puede acceder mediante dispositivos de diferente naturaleza (smartphones, ordenadores y tablets) mediante la inclusión de un módulo Bootstrap que facilita la creación de vistas adaptables en función del tamaño de la pantalla del dispositivo.
- Se ha logrado un sistema capaz de gestionar el ciclo de vida de cada una de las incidencias, desde su creación hasta su resolución, pudiéndose establecer un seguimiento de las mismas por parte de un determinado tipo de usuario (mánager).
- Se han implementado diferentes tipos de usuario (cliente, técnico y mánager) para la aplicación gracias a la correcta implementación y diseño de una serie de patrones de diseño (Modelo – Vista - Controlador). Dada la estructura empleada, se pueden añadir más tipos de usuario en un futuro de una forma sencilla.
- Se ha conseguido minimizar la complejidad del programa a la hora de lograr la portabilidad entre diferentes dispositivos al permitir que la funcionalidad sea alojada en un servidor externo, así como gracias a la inclusión del framework Bootstrap.
- Se ha logrado una aplicación flexible, no dedicada al ámbito puramente de IT, permitiendo libertad a la hora de crear los diferentes grupos de técnicos, secciones y centros.
- Por medio de una serie de patrones arquitectónicos de software bien documentados y reconocidos por la comunidad se ha logrado un producto fácil de mantener y extender o ampliar.

8.2 TRABAJO FUTURO

Se definirán y explicarán las futuras líneas de trabajo posibles para un hipotético escenario futuro próximo de mejora continua del producto en esta parte del documento.

8.2.1 Integración con redes sociales

La integración con las Redes Sociales (Facebook, Twitter, Google+) es un aspecto que muchas de las aplicaciones similares estudiadas no incorporan en sus productos. Esto puede ser una vía de mejora interesante en un mundo donde estos productos tienen cada vez más presencia; al dar funcionalidades de integración que ofrezcan comodidades al usuario, como, por ejemplo, hacer login sin necesidad de introducir una contraseña, compartir eventos en las Redes Sociales o realizar un seguimiento de las incidencias a través de ellas.

De las aplicaciones estudiadas, la única que incorporaba la integración con redes sociales (Google+), era ZenDesk [2], y tan sólo para dar la posibilidad de hacer login a través de ella.

8.2.2 CMDB, base de datos de la gestión de configuración

Una opción interesante es integrar esta herramienta con un gestor CMDB (base de datos de la gestión de configuración). Esto es una base de datos que contiene detalles relevantes de cada CI (ítem/elemento de configuración) y de la relación entre ellos, incluyendo el equipo físico, software y la relación entre incidencias, problemas, cambios y otros datos del servicio de TI. La CMDB es un repositorio de información donde se relacionan todos los componentes de un sistema de información, ya sean hardware, software, documentación,

etc. De esta forma se podría tener un control de los servicios proporcionados, así como un control sobre el inventario disponible, instalaciones, etc…

Esta es una opción muy interesante ya que integra dos aspectos muy interesantes para todas las compañías: un sistema de notificación de incidencias y peticiones, junto con un sistema para gestionar los bienes que esta dispone desde un mismo producto.

8.2.3 Sistema de penalizaciones

Al ser un sistema tan abierto a un número grande de usuarios, es necesario contemplar un sistema para evitar los posibles abusos de los mismos: creaciones masivas de tickets no relevantes, insultos o lenguaje inapropiado, uso fraudulento de la aplicación. Este sistema podría basarse en sanciones a los usuarios que abusan de la aplicación (ejerciendo un control sobre los tickets que se suben a la base de datos, sistemas de logs, o sistemas de reportes entre usuarios).

8.2.4 SMS

En el estado actual de la herramienta, no hay forma de notificar a un usuario eventos como la resolución de un ticket abierto, o la asignación de un ticket a un técnico si no es a través del correo electrónico o en la propia página. Una solución eficaz podría ser la integración con sistemas SMS para notificar de estos sucesos a los usuarios interesados en los mismos y así dar mayor flexibilidad al usuario a la hora de recibir este tipo de notificaciones.

8.3 VALORACIÓN PERSONAL

Este Trabajo de Fin de Grado ha sido afrontado como un reto personal originado por una idea propia y gracias a la aceptación y las correcciones del tutor de dicho proyecto se ha conseguido realizarlo de una forma satisfactoria. Pese a haber elaborado una planificación que un principio se pensó que sería incluso holgada, la finalización del mismo ha supuesto más horas y trabajo de los previstos debido a cambios en la estructura de la memoria (el presente documento) y el perfeccionamiento de la propia aplicación, minimizándose las situaciones de error, pero a día de hoy puede indicarse que el esfuerzo invertido ha merecido la pena.

Se ha conseguido elaborar un trabajo con espíritu crítico, analizando las ventajas e inconvenientes de cada decisión; detallista, dando especial importancia a las justificaciones de las decisiones tomadas y analizando en mayor o menor medida alternativas a las que se han utilizado. Todo esto ha ayudado mucho a comprender el verdadero significado que un proyecto de ingeniería tiene. La elección de emplear determinadas tecnologías frente a otras que podrían haber resultado más cómodas de implementar, ya que se han empleado soluciones que no se han visto en ninguna asignatura del Grado como pueden ser AJAX, PHP, Bootstrap o jQuery; ha supuesto un grado de madurez añadido para mi experiencia. Ha sido gratificante el hecho de haber podido comprender el funcionamiento de determinadas soluciones, así como la integración con otros sistemas por uno mismo.

Se ha de añadir, además, que se han experimentado momentos de frustración y cierto desamparo al tratar de enfrentarme a tecnologías nunca antes vistas. La inmensa cantidad de información que la red proporciona puede ser un arma de doble filo: en ocasiones puede resultar caótica, no ajustarse a las necesidades de uno, o incluso ser incorrecta. Aquí es donde entra de nuevo en juego el espíritu crítico para evaluar si la información a la que se está accediendo es la adecuada o no. Por otra parte, la documentación oficial de una determinada tecnología puede resultar abrumadora dadas las dimensiones de la misma, así como demasiado técnica, requiriéndose una comprensión en mayor o menor medida de la tecnología en su totalidad. Ha sido frecuente la necesidad de una función determinada de una tecnología o la utilización de un depurador adecuado, lo que ha supuesto el hecho de tener que indagar a lo largo de las diferentes soluciones que se proporcionan. Pese a los trastornos sufridos, esto puede ser en definitiva lo que más haya servido para mi evolución educativa ya que se han descubierto productos hasta el momento desconocidos o se han probado diferentes implementaciones para un mismo problema hasta dar con el más adecuado; y la experiencia personal me dice que en muchos de

los casos se aprende más de los errores que de los aciertos, pese a los ratos de frustración, que terminan siendo algo pasajero.

9 ANEXOS

9.1 DEFINICIONES Y ACRÓNIMOS

- Servicio de Soporte: Servicio que se ofrece desde una parte técnica que sabe dar salida a una determinada situación hacia una parte solicitante.
- HelpDesk: conjunto de recursos tecnológicos y humanos, para prestar servicios con la posibilidad de gestionar y solucionar todas las posibles incidencias de manera integral, junto con la atención de requerimientos relacionados a las Tecnologías de la Información y la Comunicación (TIC).
- Ticket: Abstracción de una petición de servicio, en la cual se encapsula el solicitante, el técnico asignado a dicha solicitud y la solicitud en sí.
- Usuario final: Usuario que recibirá el servicio.
- Técnico: Persona cuya función es la de resolver incidencias en un entorno HelpDesk.
- Mánager: Persona cuya función será llevar un control sobre las incidencias que se crean en el sistema, así como la responsabilidad de asignar estas incidencias entre los diferentes técnicos.
- SLA: (Service Level Agreement) contrato escrito entre un proveedor de servicio y su cliente con objeto de fijar el nivel acordado para la calidad de dicho servicio. El SLA es una herramienta que ayuda a ambas partes a llegar a un consenso en términos del nivel de calidad del servicio, en aspectos tales como tiempo de respuesta, disponibilidad horaria, documentación disponible, personal asignado al servicio, etc.
- Métrica: Medida o conjunto de medidas destinadas a conocer o estimar el tamaño u otra característica de un software o un sistema de información, generalmente para realizar comparativas o para la planificación de proyectos de desarrollo.
- IT: Siglas de Information Technology: Tecnologías de la información.
- CMDB: Siglas en inglés de Base de Datos de la Gestión de Configuración.
- SGDB: Siglas de Sistema de Gestión de Bases de Datos. Conjunto de programas dedicados al almacenamiento de información en bases de datos, realizan tareas de optimización, análisis, o generación de informes.

9.2 MANUAL DE USUARIO

Para mostrar el funcionamiento de la aplicación se detallará el proceso paso a paso del ciclo de vida de un ticket, desde su creación, hasta su cierre.

La primera pantalla que aparecerá en la aplicación será la de login, solicitando un usuario y una contraseña. En el campo de usuario:

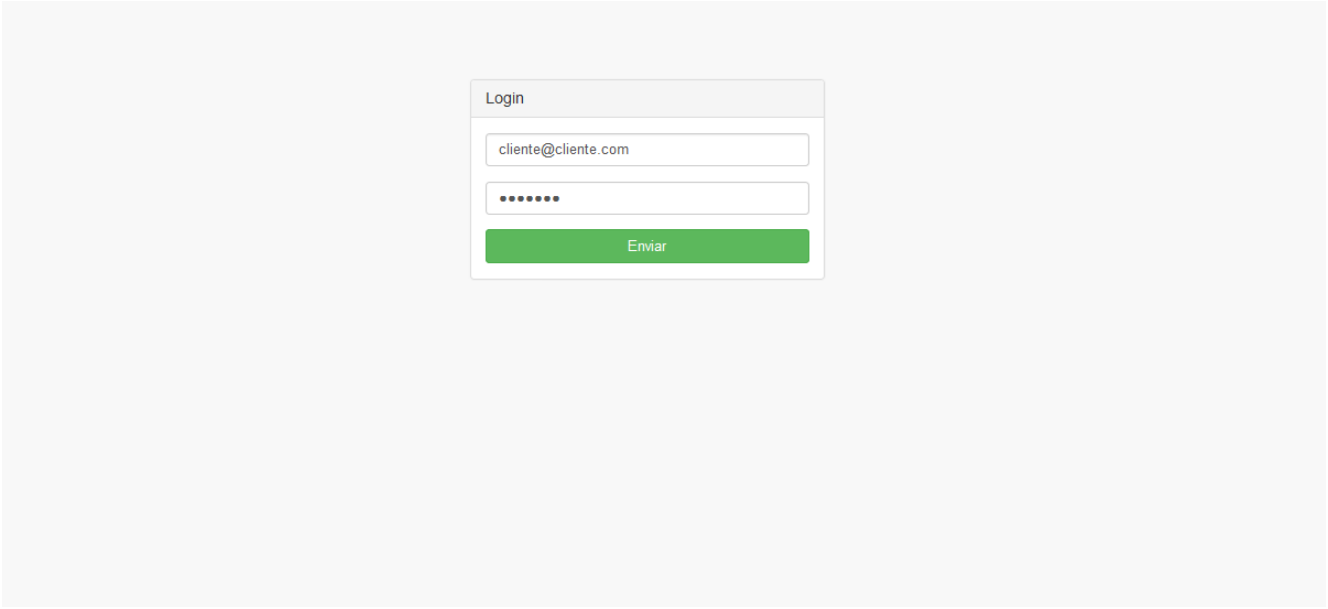


Figura 70: Pantalla de login

Una vez dentro de la aplicación (si el proceso de autenticación ha sido satisfactorio, en caso contrario, volverá a aparecer la pantalla de login), aparecerá una pantalla de bienvenida, junto con el diseño dedicado al tipo de usuario en cuestión. Para este manual se supondrá que el usuario que ha hecho login es de tipo cliente. La pantalla que aparecerá será la siguiente:

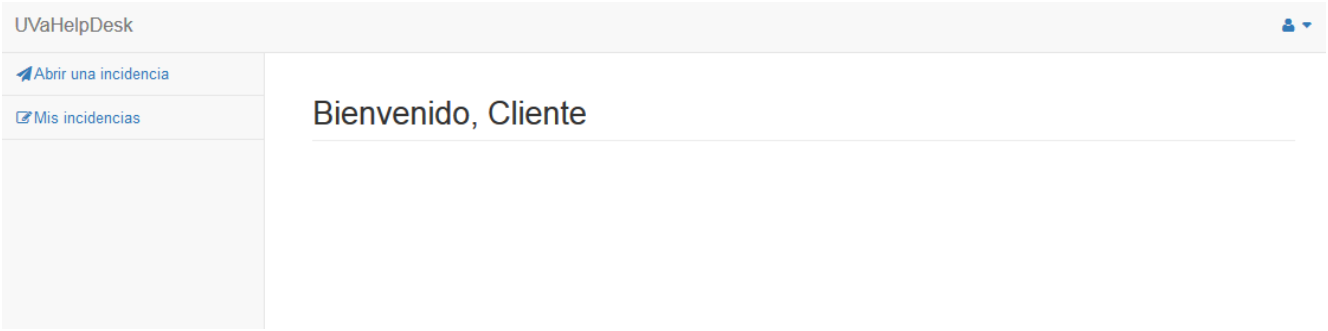


Figura 71: Pantalla de bienvenida

Para abrir una incidencia, basta con seleccionar en el menú la opción correspondiente y rellenar el formulario:

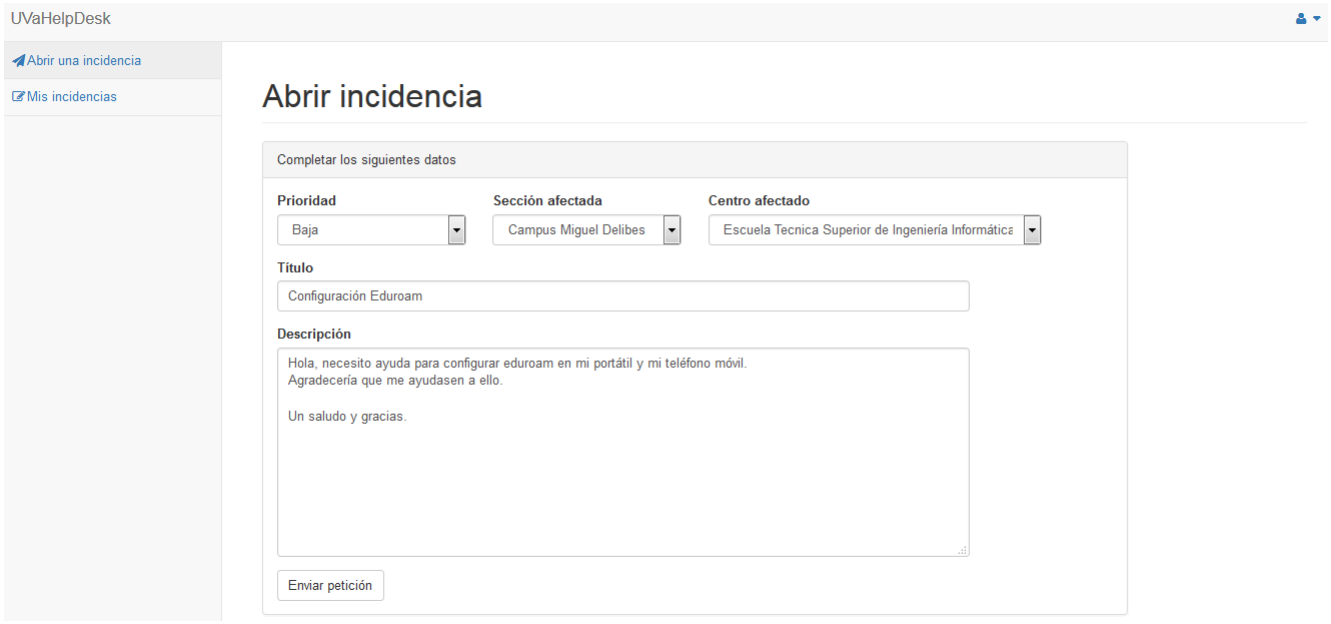


Figura 72: Formulario de creación de ticket

Una vez se envían los datos se mostrará un mensaje con el resultado de la transacción (éxito o error). Como cliente, se pueden consultar los tickets que se han creado en el sistema por dicho cliente, seleccionando la opción del menú lateral:

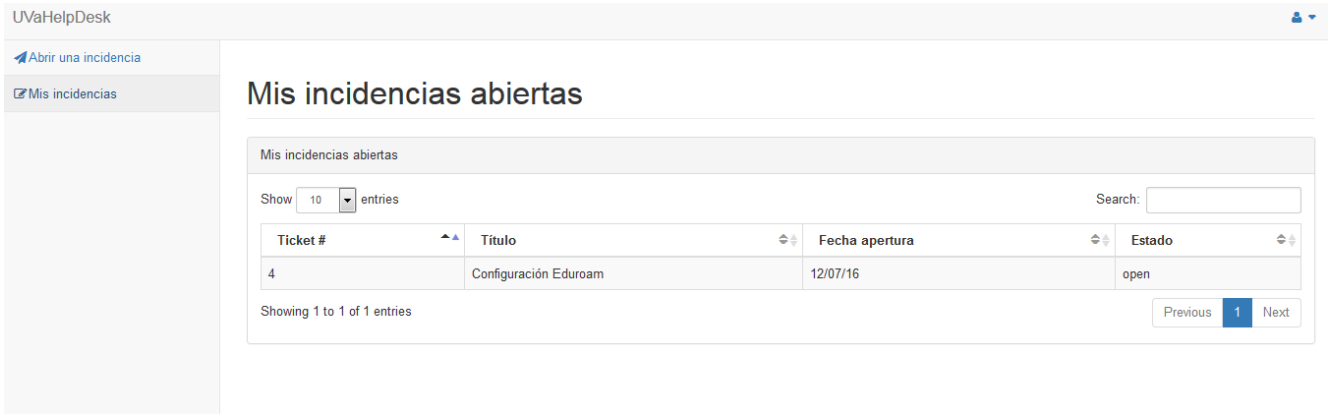
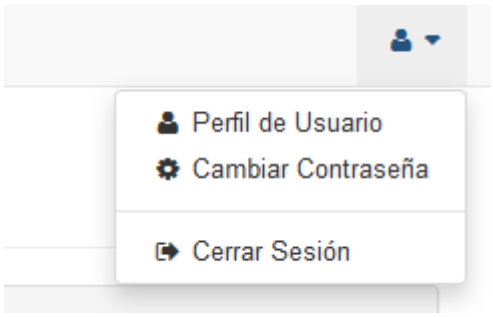


Figura 73: Listado de incidencias



El siguiente paso en el ciclo de vida del ticket, será su recepción por parte del mánager responsable de la sección en la que se ha creado. Para cerrar sesión, se puede seleccionar la opción del menú superior correspondiente, y volverá a aparecer la pantalla de login.

Figura 74: Menú superior

Se verá ahora cómo es el proceso para asignar a un técnico el ticket recién creado. Para ello habrá que hacer login como mánager.

Figura 75: Pantalla de asignación

Seleccionando en primer lugar el grupo del primer desplegable, y después el técnico de dicho grupo que se quiera asignar, se procesará la asignación. Se dará la opción de introducir un comentario que acompañará al historial de modificaciones en los detalles del ticket. Alternativamente, se puede acceder a los detalles del ticket seleccionando su fila:

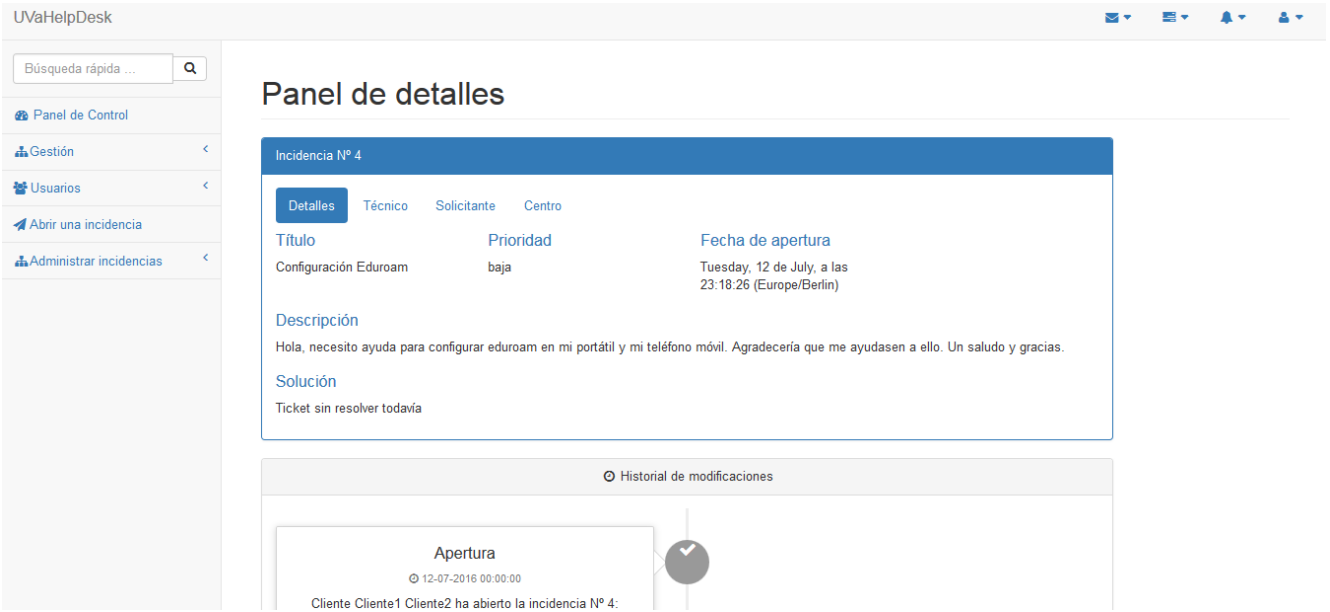


Figura 76: Detalles ticket

En esta pantalla se podrán hacer más reasignaciones a diferentes técnicos y todas ellas se verán reflejadas en el historial de modificaciones.

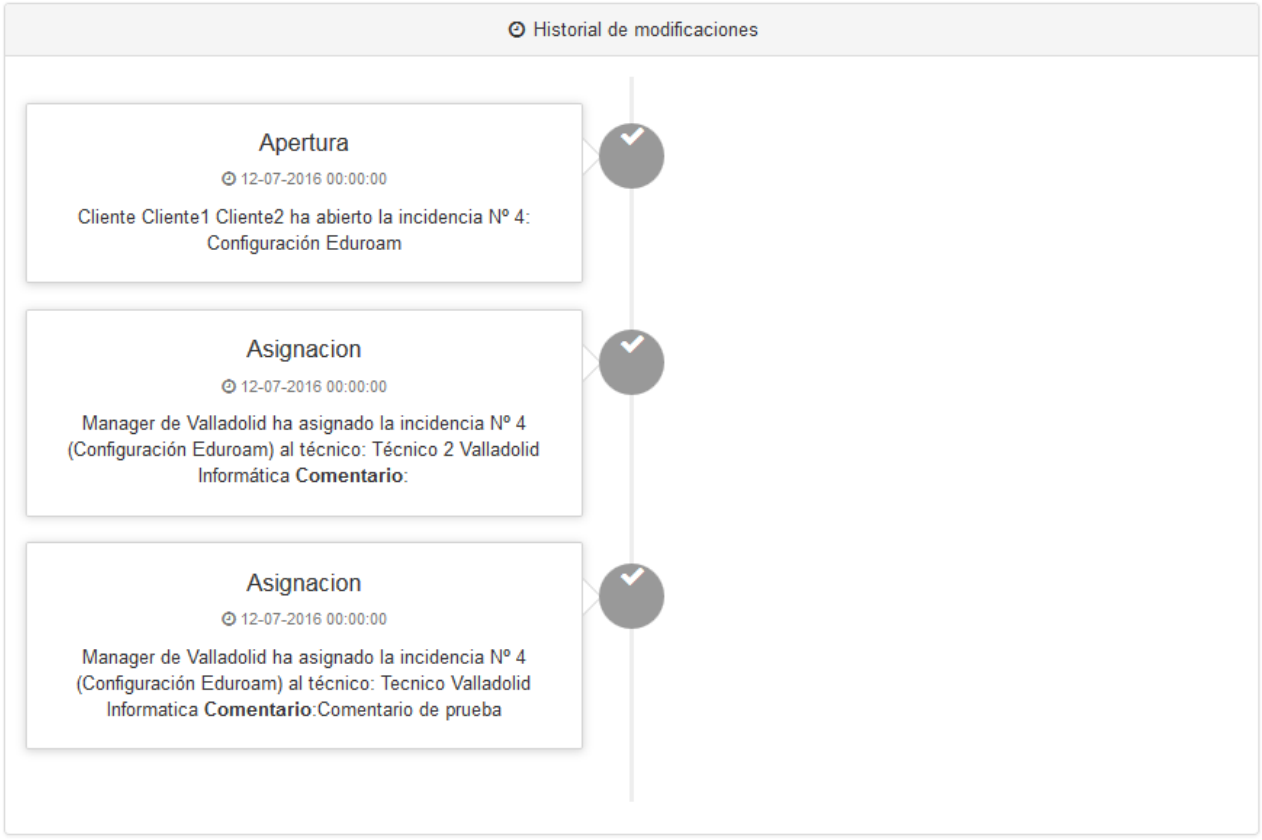
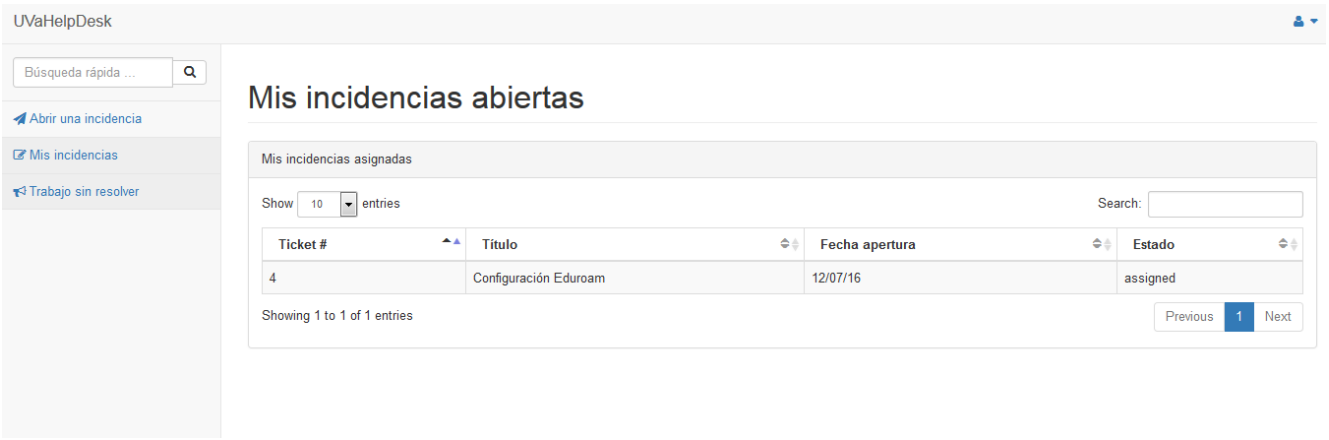


Figura 77: historial de modificaciones

El siguiente paso en el ciclo de vida del ticket es su resolución, está la hará uno de los técnicos registrados en el sistema, a través de la opción del menú lateral correspondiente:



Una vez en los detalles del ticket podrá introducir la solución que ha empleado para esta incidencia, así como un comentario, como con cada una de las modificaciones de estado que se producen en un ticket:

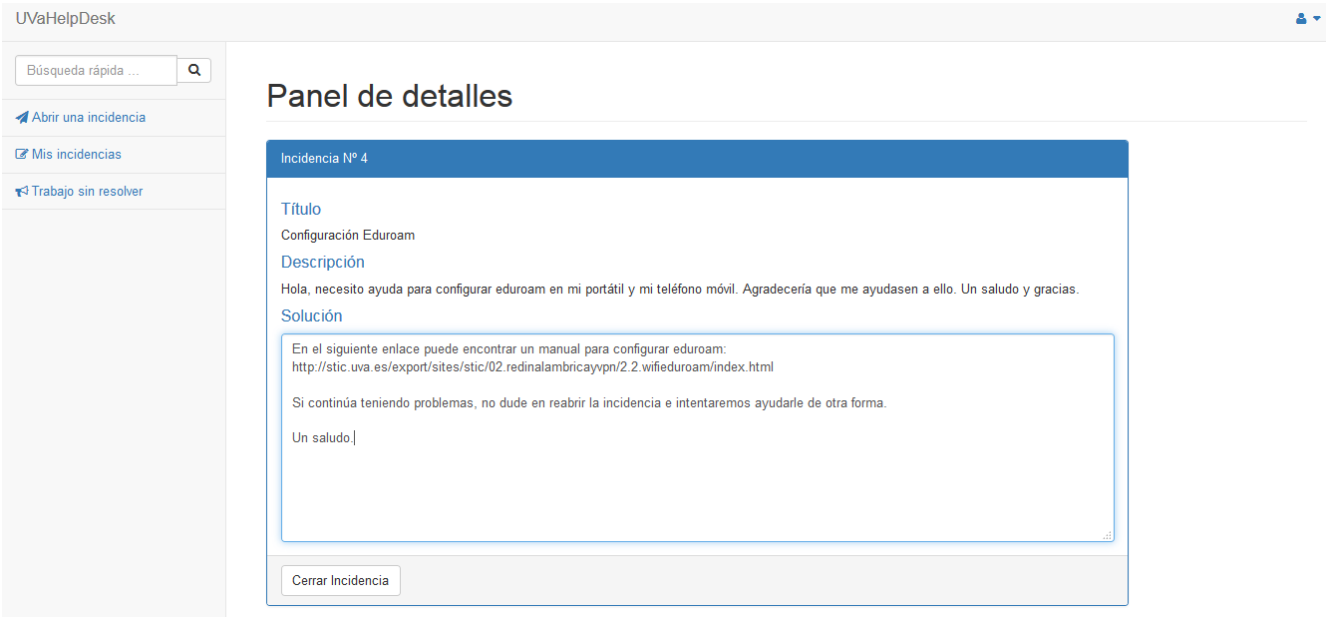


Figura 78: Detalles de ticket técnico

En este momento la incidencia habrá quedado cerrada en el sistema. Y el cliente podrá comprobar la solución que le ha proporcionado el técnico y volver a reabrir el ticket si lo considera necesario.

9.3 CONTENIDO DEL CD

El CD está estructurado con los siguientes directorios:

- Diagramas: Directorio con el fichero que contiene todos los diagramas realizados.
  - Modelos.asta: Fichero con todos los modelos de diseño.
- BD: Directorio con el contenido para crear la base de datos.
  - Uvahelpdesk_bd.png: Imagen del esquema del modelo relacional de la base de datos.
  - Uvahelpdesk_bd.sql: Script para la creación de la base de datos final.
- Doc: Directorio que contiene la memoria del Trabajo de Fin de Grado.
  - Memoria.pdf: Fichero de la memoria del TFG.
  - Planificacion.mpp: Fichero MSProject con la planificación del TFG.
- Src: Directorio con el código fuente de la aplicación.
  - Bootstrap: Directorio con el código fuente del framework Bootstrap.
  - Config: Directorio con el fichero de configuración.
  - Controladores: Directorio con el código fuente de los controladores.
  - DataMapper: Directorio con el código fuente de los DataMapper.
  - Librerías: Directorio con ficheros de utilidades funcionales.
  - Media: Directorio con el contenido multimedia de la aplicación.
  - ModeloDominio: Directorio con el código fuente del modelo de dominio.
  - Vista: Directorio con el código fuente de las vistas.
  - Index.php: Fichero con el contenido de la página inicial.

9.4 PLANTILLAS

9.4.1 Caso de Uso

Caso de Uso UC0: Plantilla

Actor Principal: *<El actor que recurre a los servicios del sistema para cumplir un objetivo>*
Personal involucrado y lista de intereses: *<Listado de las personas interesadas en el evento y por qué>*
Precondiciones: *<Lo que debe cumplirse **siempre** antes del escenario del caso de uso, se asume que son verdad (no se prueban)>*
Postcondiciones: *<Lo que debe cumplirse cuando el caso de uso se completa con éxito, debe satisfacer las necesidades del personal involucrado>*
Flujo Básico: *<Describe el camino de éxito típico, evitando bifurcaciones (estas se pospondrán para la sección Flujos Alternativos)*
Flujos Alternativos: *<Indican **todos** los otros escenarios posibles, sean de éxito o de fracaso>*
Requisitos Especiales: *<Requisitos que no son de diseño: rendimiento, fiabilidad, facilidad de uso, restricciones de diseño>*

10 BIBLIOGRAFÍA

[1] Wikipedia, «Wikipedia.org,» 24 Mayo 2016. [En línea]. Available: https://es.wikipedia.org/wiki/Acuerdo_de_nivel_de_servicio.

[2] ZenDesk, «ZenDesk,» ZenDesk, [En línea]. Available: https://www.zendesk.es/. [Último acceso: 12 07 2016].

[3] Enhancesoft, «osTicket,» Enhancesoft, [En línea]. Available: http://osticket.com/. [Último acceso: 12 07 2016].

[4] Enhancesoft, «SupportSystem,» Enhancesoft, [En línea]. Available: https://supportsystem.com/. [Último acceso: 12 07 2016].

[5] UserVoice, «UserVoice,» UserVoice, [En línea]. Available: https://www.uservoice.com/. [Último acceso: 12 07 2016].

[6] BestPractical, «RT and RTIR,» BestPractical, [En línea]. Available: https://bestpractical.com/rt-and-rtir/. [Último acceso: 12 07 2016].

[7] C. W3, «W3 - Protocols,» W3, 2016. [En línea]. Available: https://www.w3.org/Protocols/. [Último acceso: 12 07 2016].

[8] Wikipedia, «Wikipedia - HTTP,» 22 06 2016. [En línea]. Available: https://es.wikipedia.org/wiki/Hypertext_Transfer_Protocol. [Último acceso: 12 07 2016].

[9] W3Schools, «W3Schools - HTML,» 2016. [En línea]. Available: http://www.w3schools.com/bootstrap/. [Último acceso: 12 07 2016].

[10] W3Schools, «W3Schools - CSS,» 2016. [En línea]. Available: http://www.w3schools.com/css/. [Último acceso: 12 07 2016].

[11] PHP, «PHP,» [En línea]. Available: http://php.net/manual/es/index.php. [Último acceso: 12 07 2012].

[12] Wikipedia, «Wikipedia - PHP,» 03 07 2016. [En línea]. Available: https://es.wikipedia.org/wiki/PHP. [Último acceso: 12 07 2016].

[13] Oracle, «MySQL,» [En línea]. Available: https://www.mysql.com/. [Último acceso: 12 07 2016].

[14] Wikipedia, «Wikipedia - MySQL,» [En línea]. Available: https://es.wikipedia.org/wiki/MySQL. [Último acceso: 12 07 2016].

[15] Wikipedia, «Wikipedia - AJAX,» [En línea]. Available: https://es.wikipedia.org/wiki/AJAX. [Último acceso: 12 07 2016].

[16] W3Schools, «W3Schools - AJAX,» [En línea]. Available: http://www.w3schools.com/ajax/. [Último acceso: 12 07 2016].

[17] Bootstrap, «Bootstrap,» [En línea]. Available: http://getbootstrap.com/. [Último acceso: 12 07 2016].

[18] W3Schools, «W3Schools - Bootstrap,» [En línea]. Available: http://www.w3schools.com/bootstrap/. [Último acceso: 12 07 2016].

[19] J. Resig, «jQuery,» [En línea]. Available: https://api.jquery.com/. [Último acceso: 12 07 2016].

[20] P. Kruchten, The rational unified process, Boston: Addison-Wesley, 2007 .

[21] I. Sommerville, Ingeniería del Software, México: Pearson D.L., 2006.

[22] M. Fowler, Patterns of enterprise application architecture, Boston: Addison-Wesley, 2003.

[23] E. Gamma, Patrones de diseño : elementos de software orientado a objetos reutilizable, Madrid: Addison-Wesley, 2006.

[24] D. Miller, «SBAdmin 2,» BlackRock Digital, [En línea]. Available: http://startbootstrap.com/template-overviews/sb-admin-2/. [Último acceso: 22 05 2016].

[25] C. Larman, UML y patrones : introducción al análisis y diseño orientado a objetos y al proceso unificado, Madrid: Prentice-Hall, 2010.

[26] A. Formativa, «Aula Formativa,» [En línea]. Available: http://blog.aulaformativa.com/framework-responsive-alternativas-a-bootstrap/.