

MEMORIA ESCRITA DEL PROYECTO

CFGS Desarrollo de Aplicaciones Multiplataforma

CleverHelpDesk

Autor: Victor Garcia Velasco

Tutor: Mario Gago

Fecha de entrega: 09/05/2022

Convocatoria: Curso 2021-2022 - Segundo Semestre

Documentos del proyecto: https://drive.google.com/drive/folders/1nJIDvS3_f1Q-nKMYfN3bo29RS8zZbXIB?usp=sharing



Contenido

1. INTRODUCCIÓN	4
1.1 MOTIVACIÓN	4
1.2 ABSTRACT	5
1.3 OBJETIVOS PROPUESTOS (GENERALES Y ESPECÍFICOS).....	6
1.4 CONTEXTO LABORAL.....	7
2 METODOLOGÍA USADA	8
OTROS MODELOS EVALUADOS.....	8
3 TECNOLOGÍAS Y HERRAMIENTAS UTILIZADAS EN EL PROYECTO	9
4 ESTIMACIÓN DE RECURSOS Y PLANIFICACIÓN	11
5 ANÁLISIS DEL PROYECTO	14
REQUISITOS FUNCIONALES.....	14
REQUISITOS NO FUNCIONALES.....	15
DIAGRAMA ENTIDAD-RELACIÓN	15
CASOS DE USO	17
CU_01 LOGIN.....	18
CU_02 Consultar TICKETs	19
CU_03 Creación TICKET	20
CU_04 Gestión del TICKET	21
CU_05 Adjuntar Comentarios al TICKET	22
CU_06 Gestión de Usuarios	23
CU_07 Creación Proyectos.....	24
DIAGRAMA CLASES (CONCEPTUAL).....	25
6 DISEÑO DEL PROYECTO	26
6.1 PROTOTIPOS:	26
Formulario de Login	26
Formulario principal y Consulta de Tickets	27
Formulario consulta/gestión de un ticket.....	30
Formulario Gestión Usuarios.....	30
Formulario Mantenimiento Proyectos.....	32
6.2 ARQUITECTURA APLICACIÓN SERVIDOR	33
6.2.3 Organización del proyecto.....	33
6.2.4. Persistencia en base de datos	34
6.2.5 Aspectos de seguridad.....	35
6.2.6. Ejemplo de ENDPOINT.....	35

6.2.7. Emisión y recepción emails	36
6.3 ARQUITECTURA APLICACIÓN CLIENTE	37
6.3.1 Clases Principales.....	38
6.3.2 Diseño de formularios	38
6.3.3 Compilación multiplataforma	39
7 DESPLIEGUE Y PRUEBAS	41
7.1 ESPECIFICACIONES Y MANUAL DESPLIEGUE DE LA APLICACIÓN	41
7.2. PRUEBAS DE CAJA NEGRA	43
8 CONCLUSIONES	46
9 VÍAS FUTURAS.....	47
9.1. OBJETIVOS QUE SE PLANTEARON EN LA PETICIÓN INICIAL Y NO SE ALCANZARON:	47
9.2. FUTURAS MEJORAS QUE SE INCORPORARÁN EN EL SISTEMA:.....	47
10 BIBLIOGRAFÍA/WEBGRAFÍA	49
ANEXO A. GLOSARIO.....	50
ANEXO B. MANUAL USUARIO.....	51
LOGIN:.....	51
Filtrado de Tickets:.....	52
Creación de tickets:.....	52
Creación de ticket enviando un email:	53
Añadir comentarios al ticket:	54
ANEXO C. CAPTURAS DE PANTALLA.....	56
ANEXO D. GANTT FINAL DEL PROYECTO.....	58

En la normativa de proyectos vigente encontrarás una breve descripción de cada uno de estos apartados para saber qué información debes incluir en ellos

1. Introducción

Memoria del proyecto fin de ciclo Desarrollo Aplicaciones Multiplataforma de Victor Garcia Velasco. La aplicación se llama *CleverHelpDesk* y se trata de un sistema de seguimiento de incidencias/asistencias/consultas/peticiones+mejoras orientado a un equipo de desarrollo de software, aunque se podría extender a otras áreas IT. Esta aplicación se utilizará para tener un registro de incidencias/asistencias/consultas/peticiones+mejoras así como para su gestión y saber en qué estado está cada ticket así como saber quien del equipo la ha atendido.

1.1 Motivación

En mi trabajo actual (GLOBALIA HANDLING/GROUNDFORCE) estoy a cargo del desarrollo/evolutivo de varias aplicaciones corporativas, a día de hoy tenemos más de 5000 usuarios activos que utilizan alguna o varias de nuestras aplicaciones. Hace varios años realizamos un intento de adaptar la herramienta de IT que utiliza el grupo, *SERVICE-NOW*, para que realizase las funciones de este proyecto. Fue un fracaso debido a la complejidad (sobretudo de utilización para el usuario), el problema es que esta aplicación está muy orientada a la problemática de un departamento puro de sistemas: peticiones (permisos FW, petición servidores, instalaciones, etc), gestión de dispositivos/inventarios (ordenadores, móviles, impresoras, etc) y flujos de autorización en las peticiones. Para nuestra problemática concreta realizamos una configuración compleja que no se adaptaba a nuestras necesidades y nos ralentiza mucho el trabajo de asistencia específico de nuestros productos. Tampoco nos permitía obtener algunos reportes claros donde poder comprobar, por ejemplo, cuantas peticiones recibimos por mes y de que temática. Y otro gran problema de *SERVICE-NOW* es el precio de la licencia por agente, quizás este punto fue el que lo remató.

De este intento frustrado de adaptar una herramienta existente surge esta aplicación *CleverHelpDesk*. La verdad es que hace unos años que teníamos en mente hacer algún desarrollo a medida para esta gestión, pero debido a la pandemia se tuvo que parar.

Este proyecto de fin de ciclo me ha dado la oportunidad de desarrollar una primera versión de la aplicación totalmente funcional y lista para desplegar en un entorno de producción, pero siendo realistas solo será la base de un sistema mucho mayor y con mucho más recorrido (ver punto 9.2. Futuras mejoras que se incorporarán en el sistema).

1.2 Abstract

Incident/query/request control and management system for software development teams.

For each reported issue, a ticket will be created, which can be assigned to an agent for resolution. When the ticket is closed, the user will receive an email indicating this fact.

This system allows you to indicate the type of incident, priority, project, etc. This system allows you to indicate the type of incident, priority, project, etc. The application will be multiplatform, with a version for WINDOWS and another for ANDROID.

There will be three types of user profiles:

- User, will only be able to see their tickets
- Agent, will be able to see all tickets and work with them
- Admin, the same as the agent and manage user access, create projects

1.3 Objetivos propuestos (generales y específicos)

El objetivo principal es desarrollar una aplicación funcional y multi-plataforma para poder abrir “tickets” (incidencias/asistencias/peticiones/etc) por parte de los usuarios y así los agentes poder atenderlas, modificar el status, hacer comentarios, etc. La gracia es que utilizo una tecnología que me permite, a partir de un mismo código fuente, generar una aplicación nativa WINDOWS y otra aplicación nativa ANDROID (también, en un futuro se podría hacer la aplicación nativa para MacOS e IOS). Esto lo conseguimos utilizando el entorno FIREMONKEY en el IDE DELPHI.

Otro objetivo es que la aplicación utilizará patrón de arquitectura MVC para publicar un API rest donde atacará la aplicación cliente, para esto he utilizado SPRING BOOT.

Más específicamente el sistema tendrá las siguientes funcionalidades:

- Posibilidad de hacer login (requerido para poder entrar en la aplicación)
- Generación de email con el número de ticket cuando se abra y se cierre
- Posibilidad de crear un ticket enviando un email a una dirección específica
- Formulario de navegación/consulta por los “tickets” abiertos (ver los que tengo “yo” asignados, ver los de otro agente, ver los tickets cerrados, etc)
- Formulario de creación/modificación “ticket” con los campos
 - nombre
 - descripción
 - tipo (consulta, asistencia, consulta, permisos, etc)
 - área funcional/proyecto -estado (registrado, trabajando, parado, terminado) - fecha apertura -fecha último cambio
 - fecha cierre
 - agente asignado
 - usuario que reporta el ticket
 - registros comentarios (pueden haber N y junto el comentario se guardará la fecha)
- Mantenimiento Usuarios, con tres perfiles: Administrador, agente y usuario
- Mantenimiento Áreas funcionales/Proyectos

1.4 Contexto laboral

Este proyecto me ha permitido cubrir una necesidad que tenemos actualmente para atender/coordinar/controlar las peticiones/asistencias/consultas siempre referentes a nuestro software.

En el lado personal, y que indudablemente va a mejorar mi currículum he aprovechado para empaparme en todo lo posible sobre dos tecnologías que no había utilizado nunca:

- SPRING BOOT, concretamente para el desarrollo de la API REST que consume la aplicación. También he visto la facilidad para integrar componentes en la propia aplicación, por ejemplo, cuando lo he utilizado para añadir la funcionalidad de envío y recepción emails. Me ha parecido una tecnología extraordinaria y muy sólida, al principio es un poco “extraño” debido a la utilización del principio de diseño software *inversión de control* pero una vez te acostumbras ves que es una mejora sustancial que permite desacoplar las clases. SPRING BOOT me ha parecido una tecnología muy potente para acelerar la productividad y seguridad en el “lado servidor”.
- DELPHI/FIREMONKEY, sí que he utilizado DELPHI/VCL para la creación de aplicaciones gráficas complejas (p.e. sistemas de monitorización en tiempo real, etc) siempre en entorno windows. Con este paso adelante DELPHI/FIREMONKEY es otro nivel, la verdad es que actualiza el IDE sustancialmente, aunque me ha costado porque cambia el paradigma sobretodo ya que se adapta a los nuevos tiempos y a la programación multiplataforma. El problema que he encontrado principalmente es que hay poca documentación, pero después trabajarlo y entenderlo he visto que es una herramienta potentísima. Solo hay que ver su lema, un único código para múltiples plataformas, además el compilado en cada plataforma es nativo por lo que tendrá un rendimiento muy bueno, un ejemplo sería que FIREMONKEY se utiliza mucho para hacer videojuegos en plataforma ANDROID/IOS.

2 Metodología usada

Debido al tiempo limitado y a que los requisitos están claros desde un principio he decidido utilizar el *Modelo en Cascada* para la realización de este proyecto de software. En este modelo de desarrollo clásico dividimos el ciclo de vida del proyecto en estas 5 fases:



Ilustración 1. Fases Modelo en Cascada [Elaboración Propia utilizando diagrams.net]

Las principales motivaciones que me han hecho elegir este modelo son:



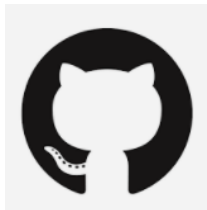

- Tener poco tiempo para realizar este proyecto (apenas 6 semanas)
- Los requerimientos ya están cerrados y claros
- Fácil de entender, planificar y realizar seguimiento

Otros modelos evaluados



En la evaluación de qué metodología iba a usar me he interesado por la utilización de las metodologías ágiles: *Scrum* y *Programación Extrema (XP)*, en ambos casos es necesario una retroalimentación por parte del cliente e ir realizando ciclos incrementales y refinando en cada uno los requisitos (tanto de ciclos anteriores como añadiendo nuevos requisitos). Evidentemente no voy a tener un cliente que me revise las entregas de cada ciclo, y la realización de varios ciclos aportando en cada entrega valor funcional al producto me habría llevado más tiempo, este ha sido mi principal motivo para descartar estas metodologías.

3 Tecnologías y herramientas utilizadas en el proyecto

Herramientas desarrollo:

Herramienta	Descripción y uso en este proyecto
	<p>IntelliJ IDEA 2021.2.3 (Community Edition)</p> <p>IDE para desarrollo de la parte servidor para la creación de un servidor <i>REST/JSON</i> al que se podrá conectar la aplicación cliente.</p> <p>El lenguaje de programación utilizado es <i>JAVA</i> junto con las librerías <i>SPRING BOOT</i>.</p>
	<p>Delphi 10.4 Community Edition</p> <p>IDE desarrollo para la parte cliente, apoyado con su librería para desarrollo multiplataforma <i>FIREMONKEY</i> . con un paradigma claro <u>un único código fuente para varias plataformas</u>. En este proyecto se creará el programa cliente para Windows y Android. También permitiría crearlo para MacOS e IOS pero es necesario tener un “Mac” para hacer la compilación y no dispongo de él.</p>
	<p>Github</p> <p>El repositorio del proyecto se alojará en Github, que a su vez utiliza Git como sistema de versionado y control de versiones.</p> <p>Ruta del proyecto Github: https://github.com/victorgv/CleverHelpDesk </p>
	<p>Postman</p> <p>Aplicación para ayudar en el desarrollo de APIs y testing.</p> <p>La utilizo como herramienta de pruebas y depuración al realizar los diferentes ENDPOINT del API.</p>

Base de datos:

Herramienta	Descripción y uso en este proyecto
	<p>Oracle 11XE</p> <p>Gestor de base de datos ORACLE en su versión XE (eXpress edition), es la edición gratuita.</p> <p>Es el motor de base de datos de la aplicación.</p>
	<p>Oracle SQL Developer</p> <p>Es el entorno de desarrollo integrado para trabajar con bases de datos Oracle que ofrece esta empresa y es gratuito.</p> <p>La he utilizado para comprobar que se crean las tablas correctamente y se cargan los datos correctamente. También lo he necesitado para probar queries.</p>

Otras herramientas de apoyo utilizadas:

- Microsoft Word, para realizar la propuesta y la memoria
- Excel, para hacer la primera estimación de horas y preparar el glosario
- Microsoft Onenote, para tomar apuntes, ideas, check-list, etc
- teamgantt.com, para representar el diagrama Gantt inicial y actualizarlo en el tiempo para tener un control del estado de las tareas
- <https://www.diagrams.net/>, para la realización de todos los esquemas, diagrama entidad-relación, diagramas de clases, casos de uso, etc.
- <https://theslidequest.com/>, para la obtención de los iconos que utiliza la aplicación. Es una página de pago por suscripción.

Equipo hardware:

- Lenovo T490 - i7 16GB RAM y SSD 1TB, con software base Windows 11
- Samsung Galaxy Note 8, con ANDROID 9 para las pruebas en dispositivo móvil
- NAS QNAP 473 – 24GB RAM, tengo un HOST DOCKER donde corro la imagen con ORACLE 11XE

4 Estimación de recursos y planificación

Para la estimación y ver el coste en horas que podría suponer el proyecto he definido las diferentes tareas en base los requisitos funcionales, análisis (punto 5 de esta memoria) y la metodología elegida. Para ser estrictamente correcto faltaría una última fase de mantenimiento, donde se podría incluir el arranque en producción, apoyo tras arranque y un posible contrato de mantenimiento temporal, no he incluido esta fase porque no se realizará.

Por otro lado también he planificado una fase adicional “otros” para las tareas de preparación de la memoria y video de presentación del proyecto.

El detalle de tareas queda así:

Fase	ID	Actividad	Estimación (horas)
Análisis	A01	Definición Requisitos y Estimación Proyecto	12
Análisis	A02	Módelado diagramas y detallado (modelo ER, diagrama clases, casos uso)	15
Diseño	D01	Especificación de funcionalidades, detalle de clases, revisión modelo BD, etc	12
Diseño	D02	Prototipos interfaces gráficas	5
Codificación	C01	BackEnd. Creación del core de la aplicación spring boot	17
Codificación	C02	BackEnd. Clases para el modelo de datos	8
Codificación	C03	BackEnd. Controller	7
Codificación	C04	BackEnd. Servicio y lógica negocio	15
Codificación	C05	BackEnd. Bean Envío emails	5
Codificación	C06	BackEnd. Bean Recepción emails y procesado	13
Codificación	C07	Cliente. Core a la aplicación multiplataforma (windows / android)	10
Codificación	C08	Cliente. Login	5
Codificación	C09	Cliente. Formulario consulta "tickets"	14
Codificación	C10	Cliente. Formulario detalle/creación/modificación ticket. Funcionalidad eliminar. Gestión del ticket. Adjuntar/consultar comentarios. Ver histórico.	29
Codificación	C11	Cliente. Gestión Usuarios	5
Codificación	C12	Cliente. Gestión Proyectos	5
Pruebas	P01	Realización pruebas de verificación y validación	15
Otros	O01	Memoria (primera entrega 28 marzo) y revisión planificación	15
Otros	O02	Memoria revisión y finalización	11
Otros	O03	Creación diapositivas video	8
Otros	O04	Creación video presentación proyecto	8
Total (horas)			234

Ilustración 2. Estimación de horas inicial [elaboración propia utilizando Excel]

En cuanto a planificación de recursos humanos y de tiempo, yo seré el único trabajador dentro de este proyecto por lo que todas las tareas recaerán sobre mí. No contemplo coger días de vacaciones, ni festivos ya que el calendario es muy ajustado, he estimado que necesitare invertir unas 25 horas semanales (para simplificar tomaremos 5h L-V) empezando el lunes 7 marzo y terminando el 9 mayo.

Semana	Horas trabajo
07-mar	25
14-mar	25
21-mar	25
28-mar	25
04-abr	25
11-abr	25
18-abr	25
25-abr	25
02-may	25
Total (horas)	225

Ya podemos ver un ligero desvío entre las horas previstas (234h) y las horas efectivas que podré realizar (225), este pequeño descuadre lo intentaremos corregir añadiendo algún día extra de trabajo (sábado o domingo).

Y para completar esta planificación he estimado una fechas y dependencias entre tareas para que sean abordadas, las plasmo en el diagrama Gantt a continuación para tener una foto inicial del proyecto. Este dia diagrama lo voy actualizando de forma que al finalizar el proyecto podremos comparar como queda y que desviaciones hemos tenido.

Diagrama Gantt Inicial:

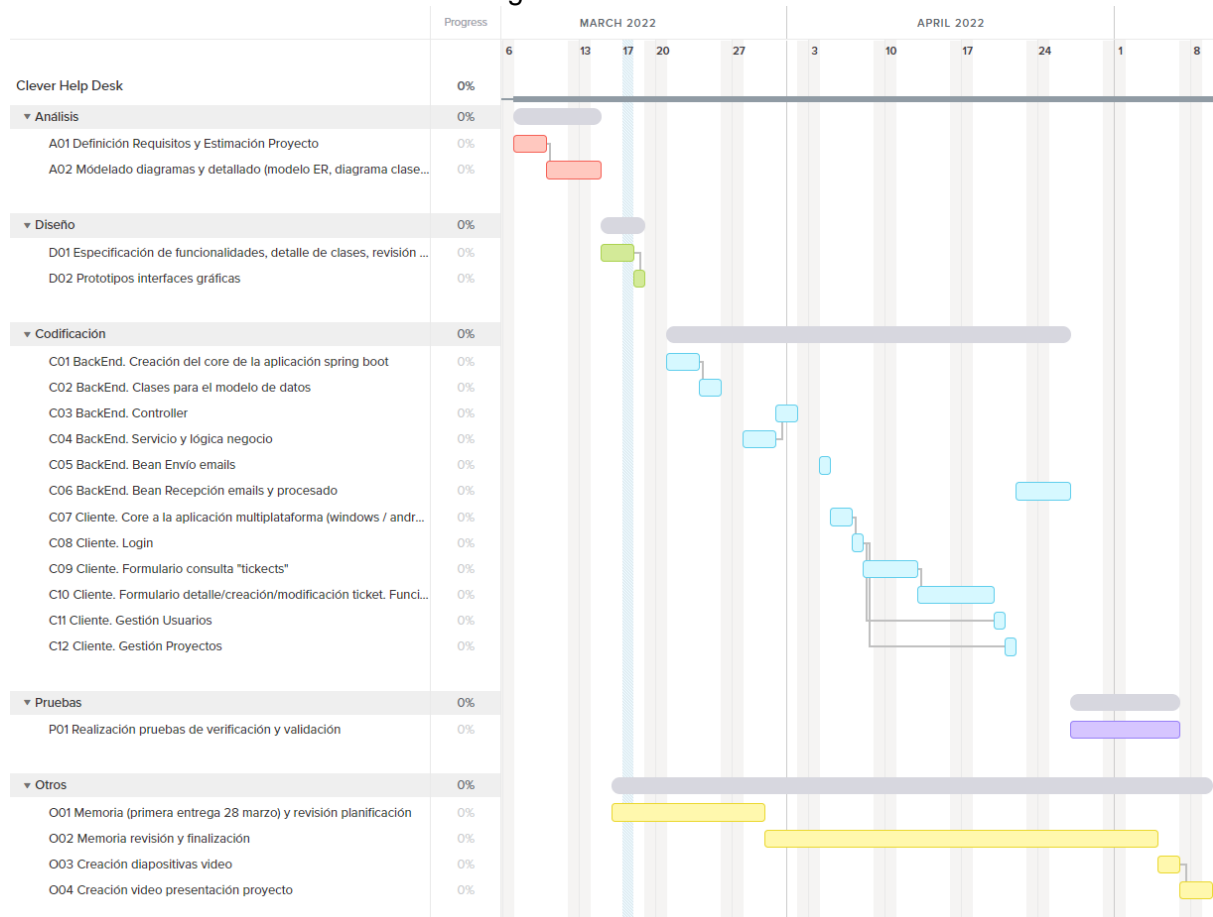


Ilustración 3. Gantt inicial del proyecto [Elaboración propia utilizando teamgantt.com]

Se puede ver el Gantt actualizado al finalizar el proyecto en el ANEXO D.

5 Análisis del proyecto

En el proceso de análisis especificamos las características del sistema, se detalla el interface que se desarrollará y se indican las restricciones de este.

Empezaremos plasmando los requisitos funcionales y no funcionales, para posteriormente plasmar los diferentes diagramas que nos facilitarán el entendimiento conceptual, así como ver las relaciones e iteraciones entre los diferentes componentes que conforman el sistema.

Requisitos funcionales

1. Se deberá hacer LOGIN para poder utilizar la aplicación CLIENTE.
2. Hay tres niveles acceso ADMINISTRADOR, AGENTE, USUARIO, se deberá tener el acceso coherente al nivel asignado.
3. Se podrán consultar los tickets según el nivel acceso:
 - a. ADMINISTRADOR y AGENTE, pueden ver todos.
 - b. USUARIO, solo los tickets que haya abierto.
4. Creación tickets.
5. Asignar un agente a un ticket.
6. Cambiar estado/proyecto/tipo ticket.
7. Poder adjuntar comentarios de texto al ticket.
8. Guardar un pequeño histórico cronológico del ticket, cuando se creo/cambios estados/etc.
9. Generación de email con el número de ticket cuando se abra y se cierre.
10. Posibilidad de crear un ticket enviando un email a una dirección genérica de la aplicación.
 - a. Solo creará el ticket si la dirección de email del remitente está registrada como usuario
 - b. En caso de no reconocer el email como usuario devolverá un email indicando que no se ha podido crear el ticket indicando que se debe poner en contacto con el administrador
11. Poder dar de alta/baja usuarios (mantenimiento de usuarios)
12. Poder crear proyectos para agrupar los tickets por temática (mantenimiento de proyectos)

Requisitos no funcionales

1. Poder utilizar la aplicación en S.O. Windows y Android mediante aplicación nativa, realizada con DELPHI/FIREMONKEY
2. Seguridad básica, guardado passwords encriptados y utilizar JWT para autenticación en el servidor REST
3. Creación servidor REST/JSON en JAVA utilizando el framework SPRING BOOT
4. Como BBDD se utilizará ORACLE, pero el lado servidor utilizará el estándar JPA con su implementación HIBERNATE de forma que permita la migración a cualquier otro gestor de BBDD relacional que lo soporte

Diagrama Entidad-Relación

A continuación se plasma el modelo entidad-relación para facilitar la representación de datos, entidades y relaciones que formarán la base de datos.

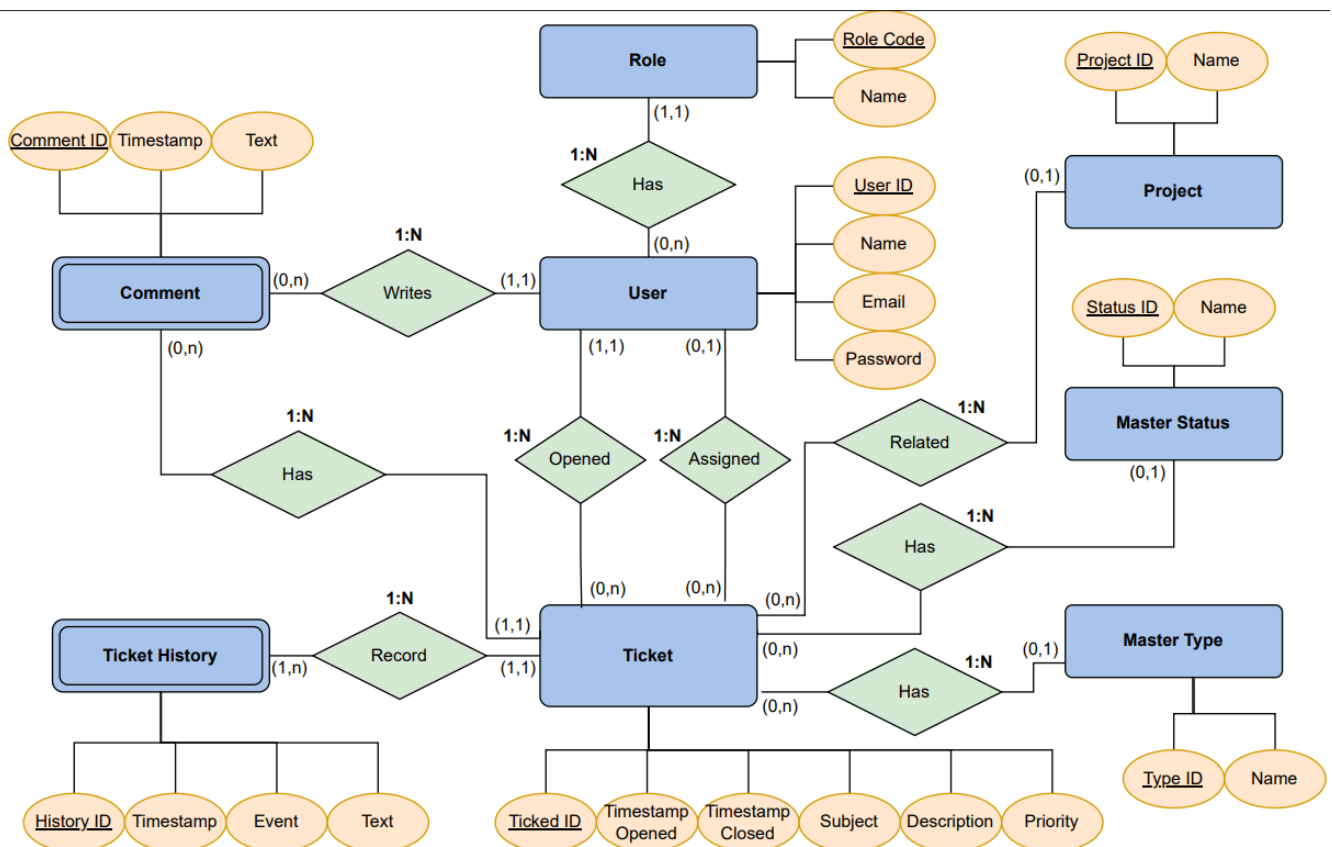


Ilustración 4. Diagrama Entidad-Relación [Elaboración propia utilizando diagrams.net]

Entidades:

- **User**, usuarios que tendrán acceso a la aplicación
- **Role**, perfiles de usuario, inicialmente serán: USUARIO, AGENTE y ADMINISTRADOR, aunque permitiría añadir nuevos roles
- **Ticket**, son las peticiones/incidencias/etc que han abierto los usuarios
- **Comment**, posibles comentarios que pueda ir introduciendo el usuario que abre el ticket o el agente de lo atiende
- **Ticket History**, es un histórico cronológico que guarda los eventos más relevantes del ticket. Por ejemplo almacenará cuando se hizo un cambio de estado y quien lo hizo
- **Project**, diferentes proyectos o áreas en las que luego podremos agrupar los diferentes tickets
- **Master type**, diferentes tipos de ticket, por ejemplo: consulta, incidencia, extracción datos BD, etc.
- **Master status**, los posibles estados que puede tener un ticket.

Casos de Uso

A través de los *Casos de Uso* describiremos el comportamiento que tiene el sistema desde el punto de vista del usuario o los procesos que van a interactuar con él. A continuación detallamos cada caso de uso detectado junto con su diagrama para mejor comprensión.

Para facilitar la comprensión se muestra el diagrama Inicial que después se detalla para cada caso de uso junto:

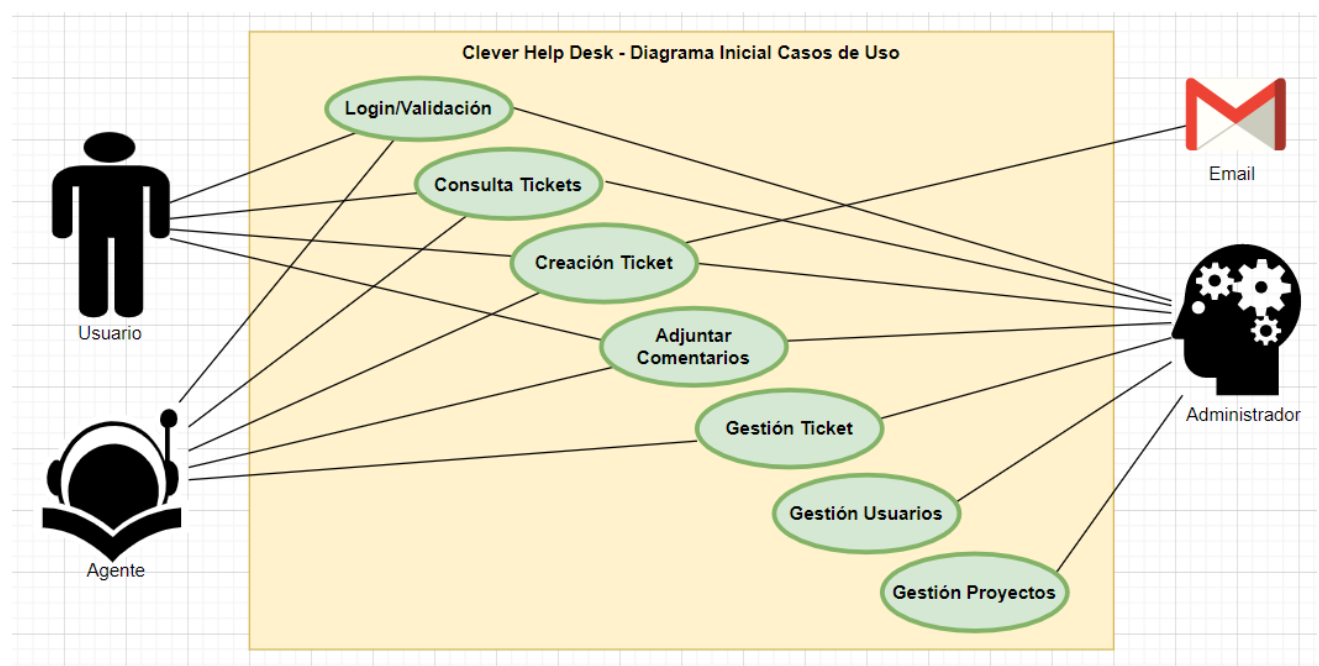


Ilustración 5. Diagrama casos de uso global [Elaboración propia utilizando diagrams.net]

CU_01 LOGIN

Identificador de Caso Uso	CU_01
Nombre	LOGIN
Descripción	Proceso de validación y acceso en el sistema
Actores	Usuario, Agente, Administrador
Secuencia normal	
Actor	Software
1. Introduce su email, password y pulsa "login"	
	2. Se valida el email/password
	3. Se accede a la aplicación
Excepciones	Software
Usuario cancela el login	Se termina la ejecución
Usuario no registrado	Muestra mensaje
Usuario/password incorrecto	Muestra mensaje
Usuario/password en blanco	Muestra mensaje y posiciona en el campo vacío
CU relacionados	
Precondición	1. Debe estar dado de alta en la aplicación. 2. Debe rellenar los campos email y password
Post condición	El usuario puede acceder al sistema

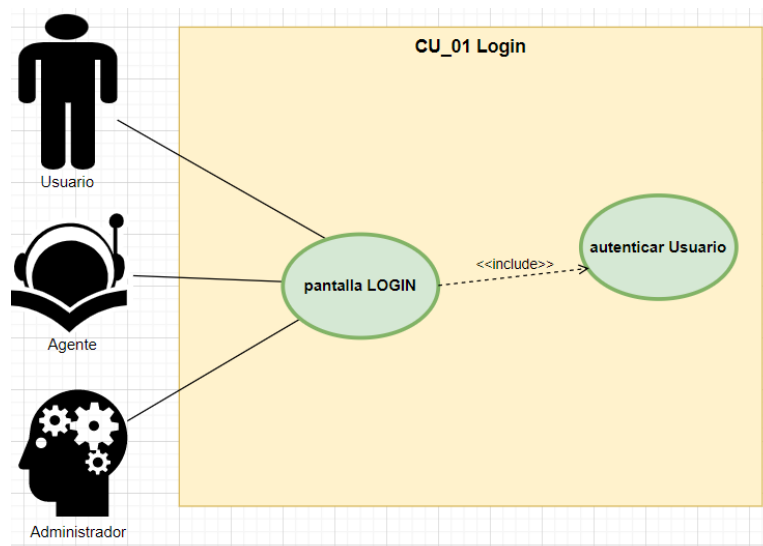


Ilustración 6. Diagrama casos de uso LOGIN [Elaboración propia utilizando diagrams.net]

CU_02 Consultar TICKETs

Identificador de Caso Uso	CU_02
Nombre	Consultar TICKETs
Descripción	Formulario principal donde se podrán buscar los TICKETs según el perfil y abrir para ver el contenido
Actores	Usuario, Agente, Administrador
Secuencia normal	
Actor	Software
1. Lanzar consulta por diferentes criterios (abiertos por mí, con un texto determinado, abiertos en una fecha X, etc)	
	2. Volcará en un GRID los TICKETs que cumplen con los criterios de consulta
3. El usuario abre un TICKET para ver su contenido	
	4. Se mostrará en una ventana o pestaña nueva para poder visualizar toda la información
Excepciones	Software
La consulta no devuelve información	El GRID se muestra sin ningún registro
Se controla el rango de fechas para no permitir consultas en rangos de fechas mayores a 366 días	El sistema muestra una ventana de aviso "el rango máximo de consulta es 1 año" y cancela la consulta
CU relacionados	CU_01
Precondición	1. debe estar registrado el email 2. estar logeado
Post condición	

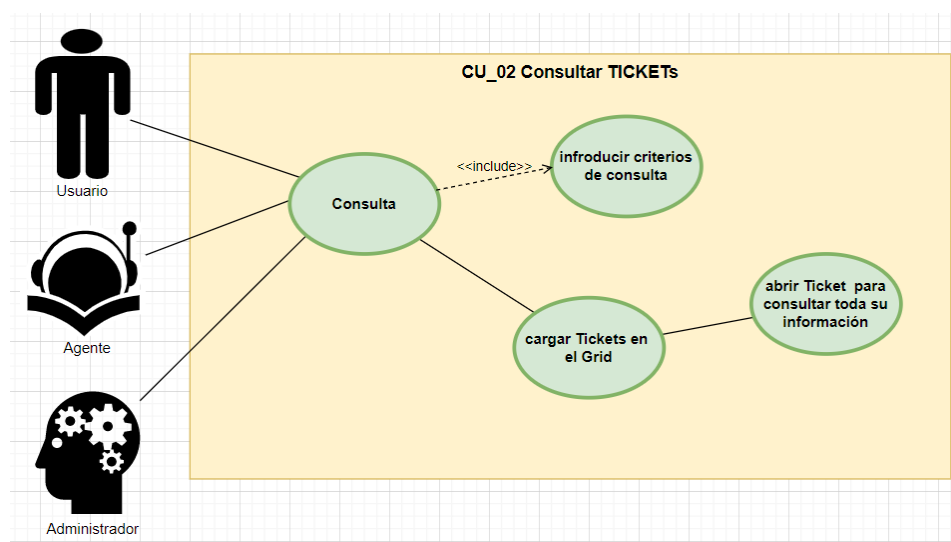


Ilustración 7. Diagrama casos de uso consulta tickets [Elaboración propia utilizando diagrams.net]

CU_03 Creación TICKET

Identificador de Caso Uso	CU_03
Nombre	Creación de un TICKET
Descripción	Proceso de creación ticket
Actores	Usuario, Agente, Administrador y Email
Secuencia normal	
Actor	Software
1a. EMAIL: se recibe un email	
1b. USUARIO: crea un ticket desde el formulario rellenando el título y la descripción de la petición	
	2. Se crea el registro, con el título y descripción
	3. Se emite un email con el número de ticket como confirmación
Excepciones	Software
EMAIL: dirección email no existe en el sistema, no está registrado como usuario	Se emite un email de respuesta indicando que no está registrando y debe ponerse en contacto con el administrador
USUARIO: cancela la creación del TICKET	No se graba el registro y se vuelve a la pantalla de consulta
CU relacionados	CU_01, CU_02
Precondición	1. debe estar registrado el email 2. Si es usuario, agente o administrador debe estar logeado
Post condición	Usuario vuelve a la pantalla de consulta tickets

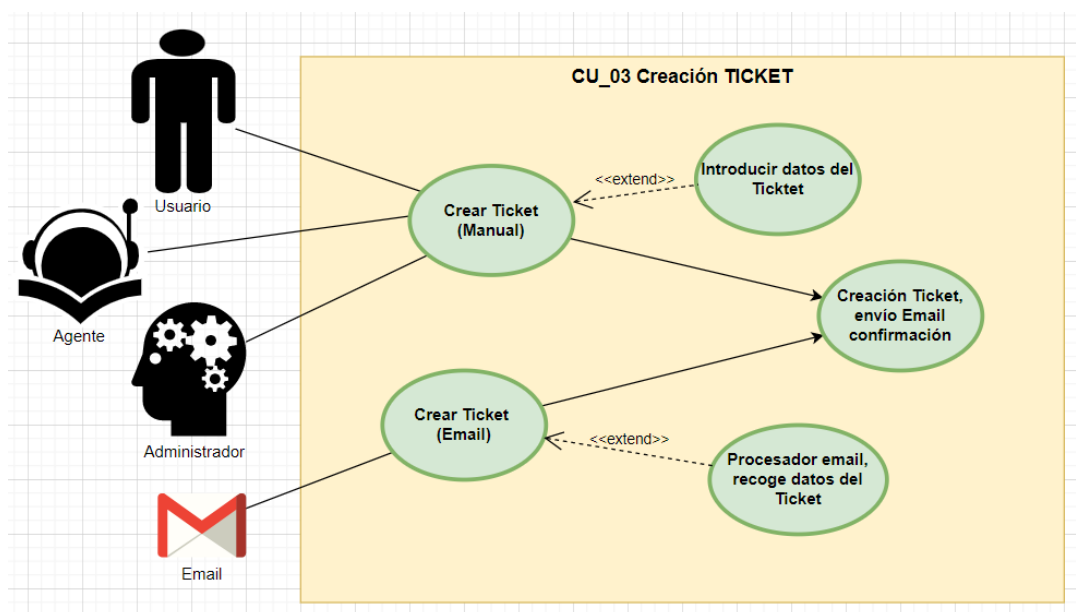


Ilustración 8. Diagrama casos de uso creación tickets [Elaboración propia utilizando diagrams.net]

CU_04 Gestión del TICKET

Identificador de Caso Uso	CU_04
Nombre	Gestión del TICKET
Descripción	Cuando un AGENTE o ADMINISTRADOR trabaja sobre el ticket, debe configurar a que proyecto pertenece y de que tipo es. Puede cambiar el estado.
Actores	Agente y Administrador
Secuencia normal	
Actor	Software
1. Acceso al formulario gestionar ticket.	
	2. Se abre el formulario
3. Se modifican los campos (agente que atiende el ticket, proyecto, tipo ticket, estado, Cerrar ticket, etc) y se pulsa "grabar"	
	4. Se confirman los cambios y se vuelve a la pantalla de consulta. En el histórico se registrará como log los cambios (por ejemplo cada vez que se cambia el estados, quien y cuando provoco la modificación)
	5. En el caso de cierre de un ticket se emitirá un email
Excepciones	Software
Intentar cerrar el ticket o pasar al estado "FINALIZADO" sin haber rellenado los campos proyecto o tipo de ticket	Se mostrará un aviso indicando que no se puede finalizar sin completar los campos indicados
CU relacionados	CU_01, CU_02
Precondición	1. el agente o administrador debe estar logeado 2. el ticket debe estar creado previamente
Post condición	Usuario vuelve a la pantalla de consulta tickets

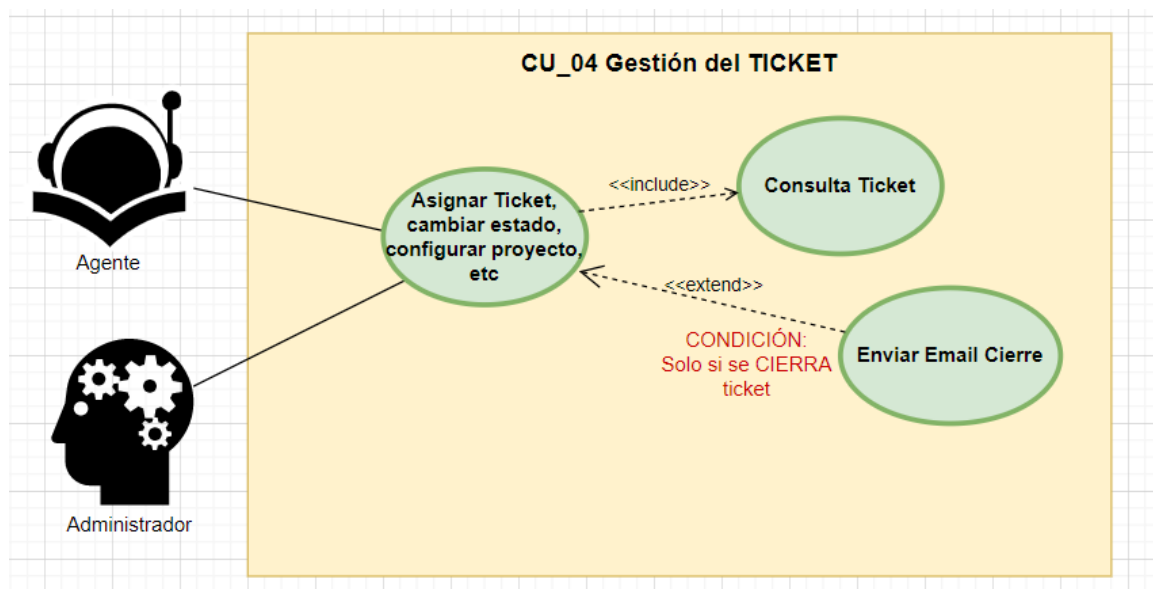


Ilustración 9. Diagrama casos de gestión ticket LOGIN [Elaboración propia utilizando diagrams.net]

CU_05 Adjuntar Comentarios al TICKET

Identificador de Caso Uso	CU_05
Nombre	Adjuntar comentarios al TICKET
Descripción	Se podrán añadir comentarios al TICKET y se podrán consultar cronológicamente
Actores	Usuario, Agente y Administrador
Secuencia normal	
Actor	Software
1. Acceso al formulario comentarios.	
	2. Se abre el formulario
3. Se introduce un comentario (debajo el último que ya hubiese) y se pulsa "grabar"	
	4. Se confirman la inserción
Excepciones	Software
Intentar introducir un comentario en blanco	Se mostrará un aviso indicando que no se puede insertar un comentario vacío
CU relacionados	CU_01, CU_02
Precondición	1. el agente o administrador debe estar logeado 2. el ticket debe estar creado previamente
Post condición	Usuario vuelve a la pantalla de consulta tickets

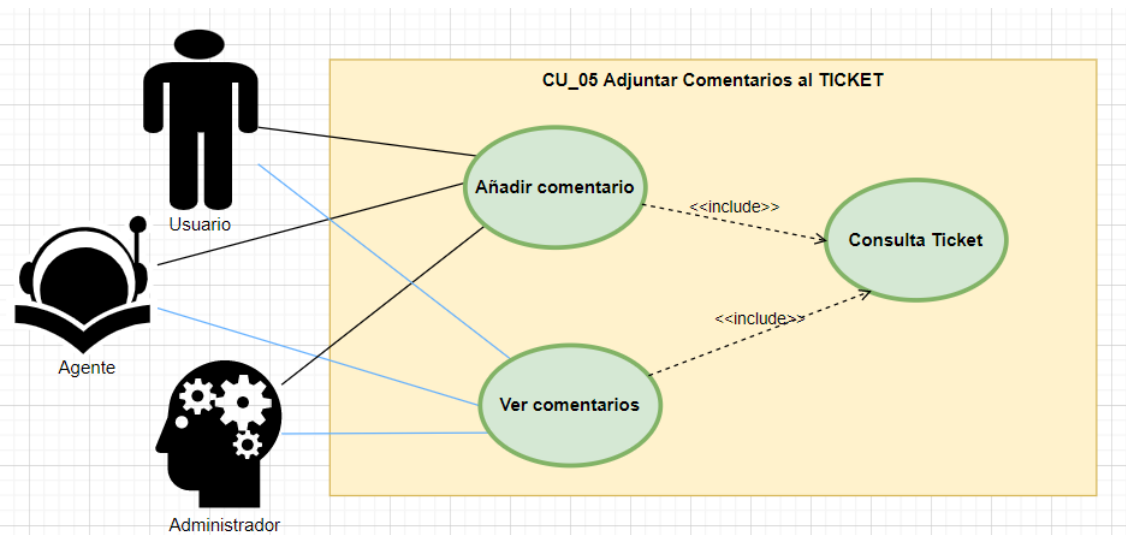


Ilustración 10. Diagrama casos de uso comentario ticket [Elaboración propia utilizando diagrams.net]

CU_06 Gestión de Usuarios

Identificador de Caso Uso	CU_06
Nombre	Gestión Usuarios
Descripción	Pequeño mantenimiento donde se podrán dar de alta y baja los usuarios en el sistema, además de otras tareas básicas como resetear el password
Actores	Administrador
Secuencia normal	
Actor	Software
1. Acceso al formulario	
	2. Se muestran los usuarios del sistema junto su perfil
3. El administrador crea un usuario, rellenando sus datos	
	4. Se confirma la inserción y el usuario puede entrar en el sistema con el password que le ha asignado el administrador
5. Se da de baja un usuario	
	6. Se confirma y a partir de ese momento el usuario no podrá acceder al sistema
Excepciones	Software
Se intenta crear un usuario que ya existe	Se mostrará un aviso indicando que no se puede duplicar el usuario
Se dejan en blanco campos obligatorios	Se avisa y se coloca sobre el campo faltante
CU relacionados	CU_01
Precondición	1. el administrador debe estar logeado 2. solo puede entrar si se tiene perfil ADMINISTRADOR
Post condición	El usuario creado puede entrar en el sistema

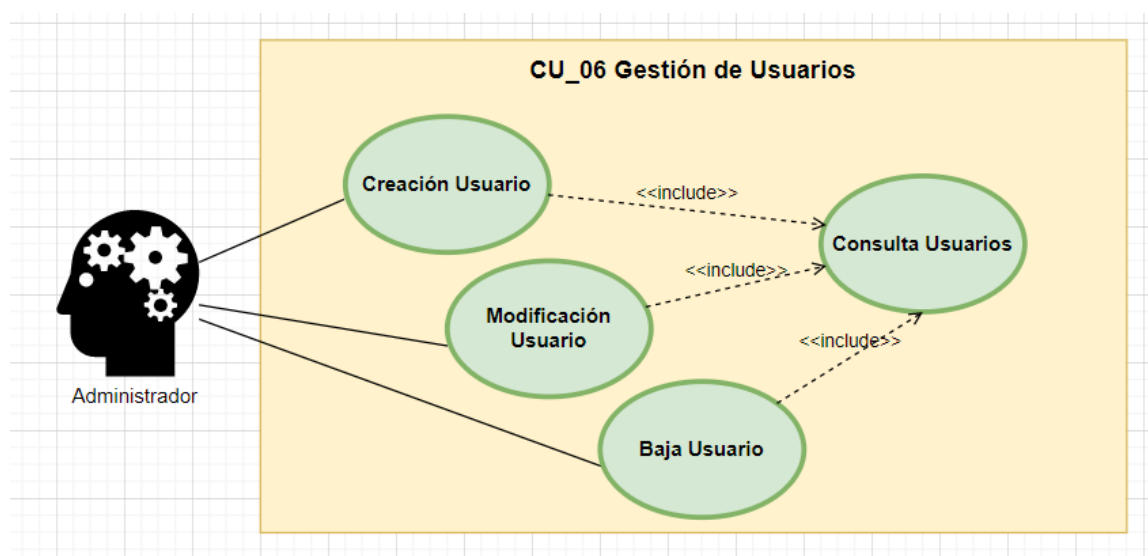


Ilustración 11. Diagrama casos de uso gestión ticket [Elaboración propia utilizando diagrams.net]

CU_07 Creación Proyectos

Identificador de Caso Uso	CU_07
Nombre	Creación Proyectos
Descripción	Pequeño mantenimiento donde se podrán dar de alta Proyectos. Estos proyectos permitirán agrupar los TICKETs en sus áreas funcionales
Actores	Administrador
Secuencia normal	
Actor	Software
1. Acceso al formulario	
	2. Se muestran los proyectos que hay creados
3. El administrador crea un proyecto, rellenando sus datos	
	4. Se confirma la inserción y a partir de ese momento se podrá utilizar para asignar en los TICKETs
5. Se modifica o da de baja	
	6. Se confirma los cambios
Excepciones	Software
Se intenta crear un usuario que ya existe	Se mostrará un aviso indicando que no se puede duplicar el usuario
Se dejan en blanco campos obligatorios	Se avisa y se coloca sobre el campo faltante
CU relacionados	CU_01
Precondición	1. el administrador debe estar logeado 2. solo puede entrar si se tiene perfil ADMINISTRADOR
Post condición	El proyecto creado ya está disponible para utilizar

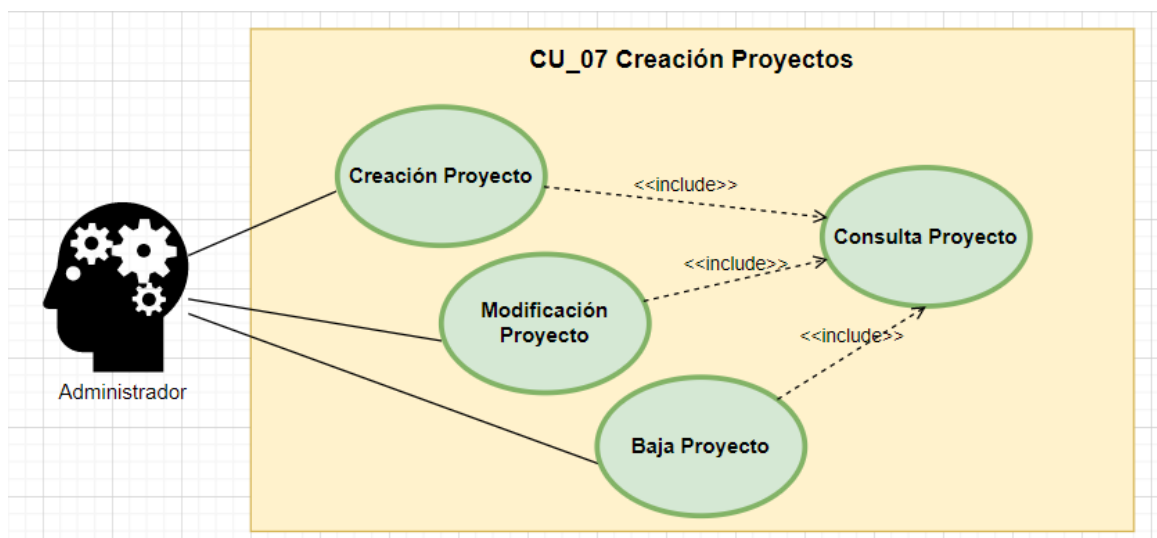


Ilustración 12. Diagrama casos de uso creación proyectos [Elaboración propia utilizando diagrams.net]

Diagrama Clases (conceptual)

En esta fase de análisis también es importante modelar un diagrama de clases para tener una aproximación mejor alcance, en este caso muestro un diagrama más conceptual ya que en la parte de diseño lo detallaré. Para facilitar el entendimiento he dividido en varios bloques, teniendo los 3 del patrón arquitectura MVC que utilizo en este proyecto más uno adicional de SERVICIOS y lógica de negocio.

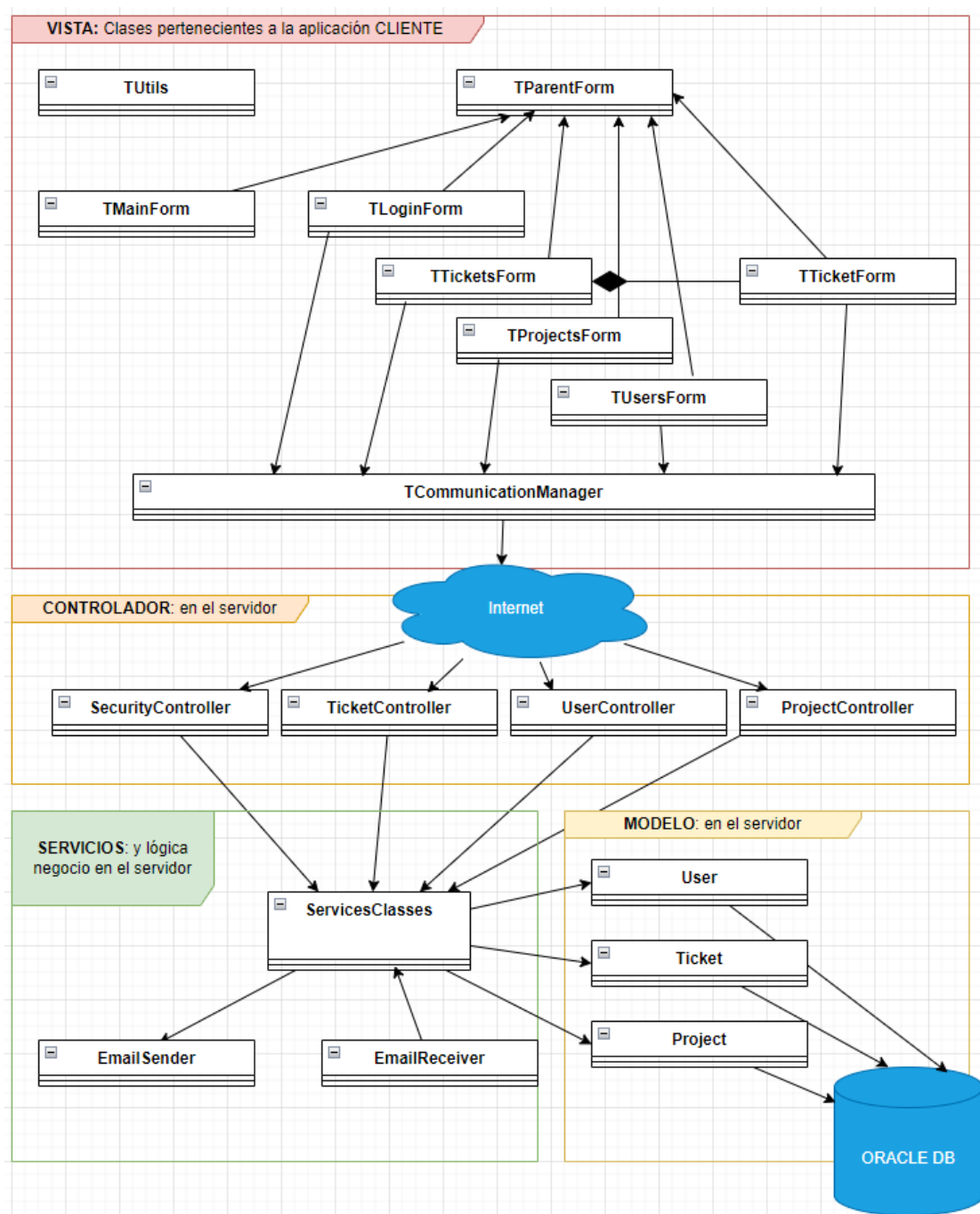



Ilustración 13. Diagrama clases [Elaboración propia]

6 Diseño del proyecto

6.1 Prototipos:

Vamos a diseñar los formularios que tendrá disponible la aplicación cliente, es aplicable tanto a la versión WINDOWS como la versión ANDROID.

Formulario de Login



The image shows a login form prototype. It has a blue header bar with the text 'Bienvenido'. Below the header, there are two input fields: 'Email' and 'Password'. The 'Password' field has a red dashed line underneath it. Below the input fields is a blue button with the text 'Log In'.

Ilustración 14. Prototipo pantalla Login [Elaboración propia]

Consta del campo Email y Password, una vez rellenados se pulsará el botón Log In para que valide y se acceda al programa si es correcto.

Formulario principal y Consulta de Tickets

Consta de dos visualizaciones: con el menú sin desplegar y con el menú desplegado.

Prototipo sin desplegar el menú:


<div>  Clever Help Desk </div>			
123	CERRADO	01/02/2022	Petición permisos accesos
22213	ABIERTO	04/03/2022	Error al generar informe personal
3234	ASIGNADO	09/03/2022	Datos incorrectos en el cálculo cierre mensual
3235	CERRADO	10/03/2022	Deadlock en el PL reinspección equipajes
3236	ABIERTO	19/03/2022	Lentitud cálculo turnos MAD
3237	ABIERTO	19/03/2022	No se reciben los télex de VY

Ilustración 15. Prototipo formulario principal de la aplicación [Elaboración propia]

En el grid de datos aparecerán 4 columnas de información que se cargarán en función de los filtros que se hayan seleccionado desde el menú desplegable.

Las columnas son:

- Identificador único del ticket
- Estado del ticket
- Fecha apertura del ticket, por espacio en el prototipo no muestro las horas pero el formato de fecha será DD/MM/YYYY HH:MI
- Título del ticket

Según el perfil del usuario podrá acceder solo a sus tickets o podrá acceder a todos los tickets abiertos.

Prototipo desplegando el menú:



VICTOR GARCIA VELASCO

Desde 01/03/2022 hasta 25/03/2022

☒ Mis tickets

Desplegable por ESTADO ▼

Desplegable por Tipo ▼




Desplegable por Proyecto ▼





Desplegable por AGENTE ▼

5256	ABIERTO	19/03/2022	Entidad calculo turnos MAD
3237	ABIERTO	19/03/2022	No se reciben los télex de VY

Ilustración 16. Prototipo desplegable formulario principal de la aplicación [Elaboración propia]

Tendremos acceso a varios elementos:

- Barra superior: indicando el nombre del usuario que ha iniciado sesión
- Barra botones:
 -  , cierra la sesión
 -  , abre el mantenimiento de usuarios (solo si el perfil es ADMINISTRADOR, si no lo es el icono estará oculto)
 -  , abre el mantenimiento de proyectos (solo si el perfil es ADMINISTRADOR, si no lo es el icono estará oculto)
- Configuración criterios de búsqueda
 - Desde 01/03/2022 hasta 25/03/2022 , para configurar el rango de fechas visible, siempre aplica sobre el campo de fecha apertura ticket
 - ☒ Mis tickets , si está marcado solo veremos los tickets que hemos abierto nosotros

-  selector para poder filtrar por un tipo de estado concreto
-  selector para poder filtrar por un tipo de ticket concreto
-  selector para poder los tickets de un proyecto
-  selector para poder filtrar los tickets asignados a un agente

Formulario consulta/gestión de un ticket

Formulario donde vamos a poder ver toda la información del ticket así como poder gestionarlo y modificarlo si tenemos perfil de AGENTE o ADMINISTRADOR.

1234 – Título p.e. Error comunicación web service			
ESTADO ▼	Abierto	25/03/2022 11:11	Cerrado 25/03/2022
Tipología ▼		AGENTE ASIGNADO ▼	
*** DESCRIPCIÓN *** se detecta en el web Service de ALTEA cuando hay más de 300 pax en los logs guarda un error buffer overflow, parece que solo aparece en vuelos de UX			
Comentarios +	VICTOR	26/03/2022	Se detecta el fallo, se corrige y prueba en desarrollo
	PABLO	25/03/2022	Cliente reclama incidencia
	VICTOR	25/03/2022	En los logs del servicio se puede obtener más información

Ilustración 17. Prototipo formulario Gestión Ticket [Elaboración propia]

Se divide en dos partes, la zona superior donde está el ID y título del ticket así como los campos necesarios para su gestión (estado, tipo, agente y descripción principal).

En la zona inferior encontramos los posibles comentarios que se hayan escrito sobre este ticket.

Formulario Gestión Usuarios

Lo dividiremos en dos formularios el primero con la lista de usuarios activos en el sistema y desde él tendremos acceso a la creación de nuevos usuarios o a la modificación de usuarios existentes. Los campos que visualizaremos en la pantalla son:

- Código de usuario único (userName)
- Nombre completo del usuario
- Email del usuario
- Perfil que tiene el usuario

Gestión Usuarios			
VICTORGV	Victor Garcia	victorgv00@gmail.com	<u>Admin</u>
MARIA	Maria Sanchez	msanchez@gmail.com	<u>User</u>
ALBERTO	Alberto Zeta	zeta@gmail.com	<u>User</u>

Ilustración 18. Prototipo formulario Gestión Usuarios [Elaboración propia]

El Formulario de creación / modificación de usuarios tendrá un diseño similar a la imagen.

Inserción/Modificación Usuario

ID	<u>UserName</u>
Nombre completo	
Perfil ▼	Email
Baja ▼	<u>Password</u>

Grabar
Cancelar

Ilustración 19. Prototipo formulario Modificación Usuarios [Elaboración propia]

Los campos disponibles en pantalla serán:

- ID único, solo aparecerá cuando se trate de una modificación
- Nombre completo del usuario
- Perfil, será un desplegable donde se podrá seleccionar el perfil que tendrá el usuario
- Email, se validará que el email introducido es correcto
- Fecha baja, se utilizará para desactivar usuarios
- Password, será un campo típico de password que oculte el contenido

Al pulsar el usuario el botón Grabar se lanzarán las validaciones y si todo es correcto se guardará en base de datos.

Formulario Mantenimiento Proyectos

La creación de proyectos y modificación se realizará desde un formulario similar a la imagen abajo indicada.

Mantenimiento de proyectos +
(SEG) Seguimiento de la Operación
(PER) Personal Handling
(CMI) Cuadro de Mando
(ASP) Asignación en Tiempo Real PASAJE
(HLD) <u>Handheld</u> Coordinación
(BRS) Sistema Reconciliación Equipajes

Ilustración 20. Prototipo formulario Mantenimiento de Proyectos [Elaboración propia]

En pantalla solo se tendrá un campo *nombre del proyecto* que será la única información que se podrá introducir/modificar.

6.2 Arquitectura aplicación servidor

Se ha creado un API/REST para la conexión de los clientes, el desarrollo se ha realizado utilizando el framework *SPRING BOOT* y como lenguaje de programación se ha utilizado *JAVA*, todo esto combinado con el *IDE IntelliJ IDEA*.

Utiliza el conocido el patrón de arquitectura Modelo-Vista-Controlador (MVC) permitiendo el desacople de la parte de datos, controlador y lógica de negocio, y la parte de vista que en este caso está fuera puesto que es una aplicación FIREMONKEY/DELPHI.

Aunque por las limitaciones de tiempo no he podido llevarlo a cabo, está pensado poder utilizarlo como microservicio y poder desplegarlo en producción utilizando contenedores *DOCKER*, asimismo la idea es que llegado a este caso se pueda correr en varias instancias, ya sea por tener una de backup o que por crecimiento de las peticiones lo requieran.

Encontraremos el código fuente de esta aplicación en el ZIP llamado *CleverHelpDesk_SOURCE.ZIP* en la carpeta *server*.

6.2.3 Organización del proyecto

La organización del proyecto se ha realizado siguiendo las recomendaciones de cualquier proyecto *SPRING*, concretamente aplicando las relacionadas con el desarrollo de APIs/REST y aplicando la lógica para tener organizado el código y que en todo momento sea fácil de mantener por cualquier persona.

Se ha organizado el código fuente en los siguientes paquetes:

- *config*, aglutina las clases de configuración y apoyo que utiliza *SPRING*
- *controller*, donde se han colocado los controladores de los endpoint que manejarán las peticiones recibirá la aplicación.
- *DTO*, son clases POJO que se utilizarán para la transferencia de datos. Nos permite enviar estructuras de datos a medida, principalmente lo utilizo para ocultar información y para reducir el volumen de información en las transmisiones.

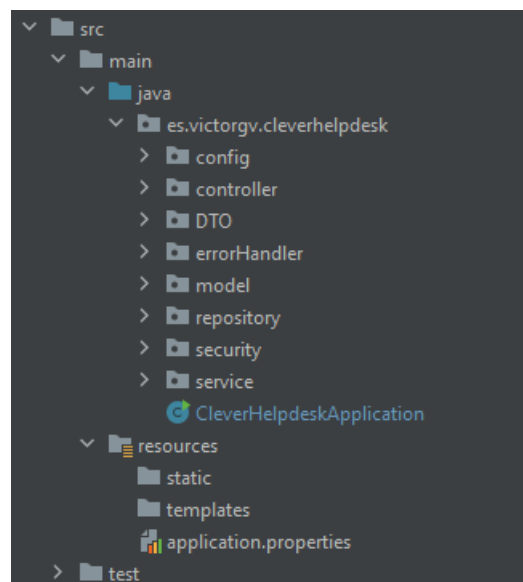
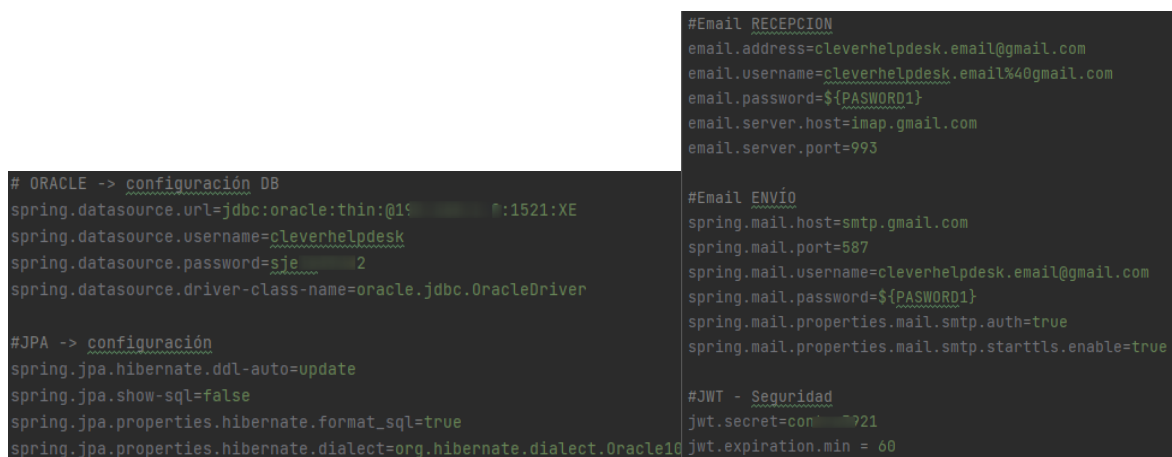


Ilustración 21. Árbol del proyecto

- *errorHandler*, aquí encontraremos la clase encargada de manejar las excepciones no controladas para poder unificar el tratamiento en un punto además de homogenizar la información y forma de elevar estas excepciones hasta los consumidores de este API.
- *model*, donde tendremos las *entidades* de la aplicación, será la representación del modelo de datos.
- *repository*, a través de interfaces nos automatiza los elementos principales del acceso a la base de datos. Junto con el *model* nos permitirá la persistencia.
- *security*, clases de configuración y aplicación de los elementos de seguridad.
- *service*, en este paquete encontraremos las clases que implementan la lógica de negocio.

La gran mayoría de parámetros de la aplicación los encontraremos en el fichero *application.properties*.



```
# ORACLE -> configuración DB
spring.datasource.url=jdbc:oracle:thin:@192.168.1.1521:XE
spring.datasource.username=cleverhelpdesk
spring.datasource.password=sje2
spring.datasource.driver-class-name=oracle.jdbc.OracleDriver

#JPA -> configuración
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=false
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.Oracle10g

#Email RECEPCION
email.address=cleverhelpdesk.email@gmail.com
email.username=cleverhelpdesk.email%40gmail.com
email.password=${PASSWORD1}
email.server.host=imap.gmail.com
email.server.port=993

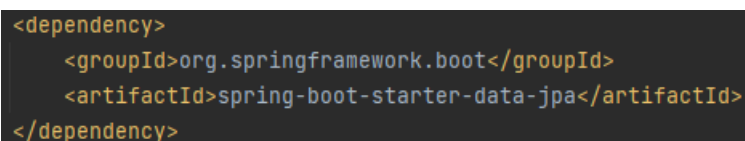
#Email ENVIO
spring.mail.host=smtp.gmail.com
spring.mail.port=587
spring.mail.username=cleverhelpdesk.email@gmail.com
spring.mail.password=${PASSWORD1}
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true

#JWT - Seguridad
jwt.secret=confecho721
jwt.expiration.min = 60
```

Ilustración 22. Fichero de configuración *application.properties* que utiliza el proyecto

6.2.4. Persistencia en base de datos

Para realizar la persistencia en base de datos se utiliza la implementación de JPA HIBERNATE que integra SPRING BOOT. Es necesario incluir en el proyecto la dependencia correspondiente:



```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
```

Ilustración 23. Dependencia necesaria para SPRING DATA

6.2.5 Aspectos de seguridad

Este proyecto se ha preparado siguiendo las recomendaciones básicas de seguridad, para ello se utiliza el estándar *JWT* en la autenticación. El usuario al validar sus credenciales contra el *API* recibirá un *TOKEN* con un tiempo de expiración (está parametrizado a 60 minutos) con ese *TOKEN* el *API* le permitirá seguir lanzando peticiones pero además sabrá que es ese usuario, y que no ha sido alterado gracias a la firma.

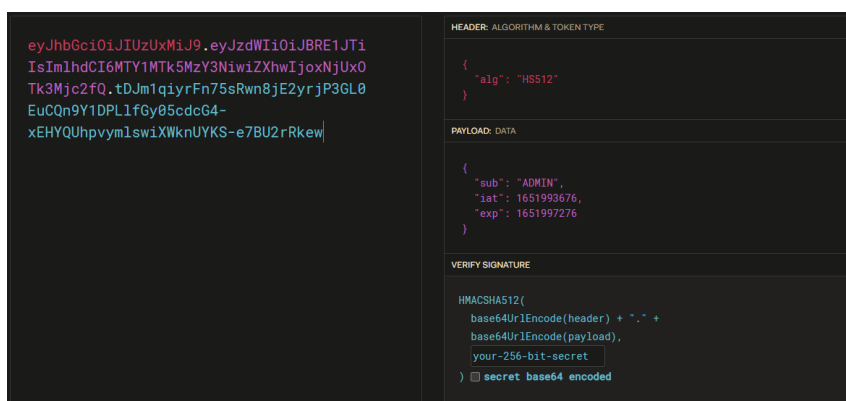


Ilustración 24. Ejemplo de decodificación *TOKEN JWT* [Elaboración propia utilizando jwt.io]

Otro aspecto importante, que no se ha podido probar en este proyecto por lo ajustado que son los tiempos, pero será el próximo paso a cumplir es la utilización de protocolo *HTTPS* para asegurar que la comunicación entre es cifrada para ello utilizaré *Let's Encrypt* que permite generar certificados gratuitos con vigencia de 3 meses autorenovables.

Para este apartado me he apoyado en el subproyecto *SPRING SECURITY*.

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

Ilustración 25. Dependencia necesaria para *SPRING SECURITY*

6.2.6. Ejemplo de ENDPOINT

Quizás sea el aspecto más importante de un *API/REST*, es el punto donde los usuarios harán sus peticiones y en función del “verbo” utilizado, el *path*, etc. El servicio será capaz de determinar que tiene que hacer y devolver.

En las respuestas se devuelven los típicos códigos *HTTP*, por ejemplo 200 para indicar OK, o 404 para indicar que no ha encontrado un recurso (NOT FOUND). Además, si lo requiere, pueden devolver en el body un bloque de información en formato *JSON*.

A modo de ejemplo vamos a ver el controlador encargado de atender a las peticiones de la entidad *User* lo encontramos en la clase *UserController*.

```
@RestController
@RequestMapping("/user")
public class UserController {
    @Autowired UserService userService;

    @PostMapping("/") // EndPoint para crear un usuario
    public User createUser(@RequestBody User_CreatedDTO nuevoUsuario) { return userService.createUser(nuevoUsuario); }

    @PutMapping("/{id}") // EndPoint para modificar el usuario con el ID pasado en el path
    public User modifyUser(@PathVariable("id") Long id, @RequestBody User_ModifyDTO modifiedUser) {
        return userService.modifyUser(modifiedUser, id);
    }
}
```

Ilustración 26. Clase UserController [Elaboración propia]

6.2.7. Emisión y recepción emails

Realmente son dos servicios diferenciados que siendo muy estrictos los podríamos haber implementado como otros dos microservicios independientes.

Respecto al envío de emails no me voy a extender, es sencillo utiliza una dependencia de SPRING y la configuración fue relativamente rápida:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-mail</artifactId>
</dependency>
```

Ilustración 27. Dependencia spring-boot-starter-mail

Recepción email:

En cambio la recepción del email fue más complicada y me llevo más tiempo del que esperaba, pero una vez entendido la forma de trabajar me parece fantástica ya que utiliza *channels* al que te puedes suscribir, un canal podría ser una cuenta de email. Pero además este sistema de *channels* / suscripción es un patrón que utiliza SPRING para muchas cosas, un ejemplo sería para poder conectarse a colas KAFKA. Para la configuración y la creación del *channel* se utiliza la clase *EmailReaderConfig* , para la suscripción y tratamiento de los emails recibidos se utiliza la clase *EmailReceiver_*.

```
@Service
public class EmailReceiver_ {

    @Autowired private UserService user_ser;
    @Autowired private ITicket ticket_repo;
    @Autowired private TicketService ticketService;
    @Autowired private IMasterStatus masterStatus_rep;

    public EmailReceiver_(DirectChannel emailDirectChannel) {
        // Nos suscribimos al canal y llamará asincrónicamente al handleMessage cada vez que entre un email nuevo
        emailDirectChannel.subscribe(new MessageHandler() {
            @Override
            public void handleMessage(Message<?> message) throws MessagingException { // Se lanza por cada email recibido
                processor(message); // Procesa el email recibido
            }
        });
    }
}
```

Ilustración 28. Clase *EmailReceiver_* junto constructor donde se hace la suscripción y se implementa el método anónimo *handleMessage* que se llamará cada vez que entre un email

Necesita la siguiente dependencia:

```
<dependency>
<groupId>org.springframework.integration</groupId>
<artifactId>spring-integration-mail</artifactId>
</dependency>
```

Ilustración 29. Dependencia necesaria para la recepción emails utilizando la implementación SPRING

6.3 Arquitectura aplicación cliente

Se ha desarrollado utilizando el IDE DELPHI y dentro de este el framework FIREMONKEY que permite el desarrollo de aplicaciones multiplataforma, el lenguaje de programación es OBJECT PASCAL.

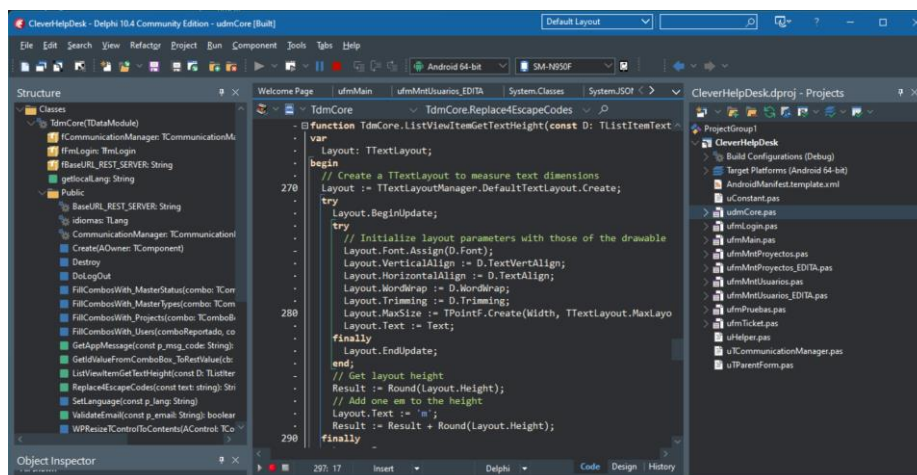


Ilustración 30. Ejemplo IDE corriendo la aplicación

La aplicación está organizada en varias unidades (cada una tiene una clase principal) y algunas de estas unidades son formularios.

6.3.1 Clases Principales

Diferenciaremos las clases en dos tipos, las de utilidad general y los formularios.

- *De utilidad general:* implementan funcionalidades que se utilizarán por toda la aplicación, las principales son:
 - TdmCore (en la unidad udmCore), implementa lógica y utilidades, por ejemplo el login (no el formulario) sino que controla si el usuario está logeado (o le ha caducado el TOKEN) de forma que pide logear. Otra utilidad es el control del “lenguaje” ya que esta aplicación está pensada para ser multi-idioma.
 - TCommunicationManager, gestiona toda la parte de comunicación. Todas las peticiones REST contra el servidor pasan por sus métodos. También implementa una lógica de control de error para detectar posibles problemas y mostrarlos de forma controlada.
- *Formularios,* los principales son estos dos.
 - TfmMain (unidad ufmMain), es el formulario principal y junto con las dos clases anteriores orquesta toda la aplicación.
 - TfmTicket (unidad ufmTicket), para poder consultar y gestionar los tickets, así como adjuntar comentarios.

6.3.2 Diseño de formularios

Para diseñar los formularios se utiliza el diseñador que incluye el IDE, y solo hay un formulario para todas las plataformas (permite crear múltiples diseños del mismo formulario para adaptarlo a las distintas resoluciones, no es lo mismo una Tablet de 10” que un móvil de 5”).

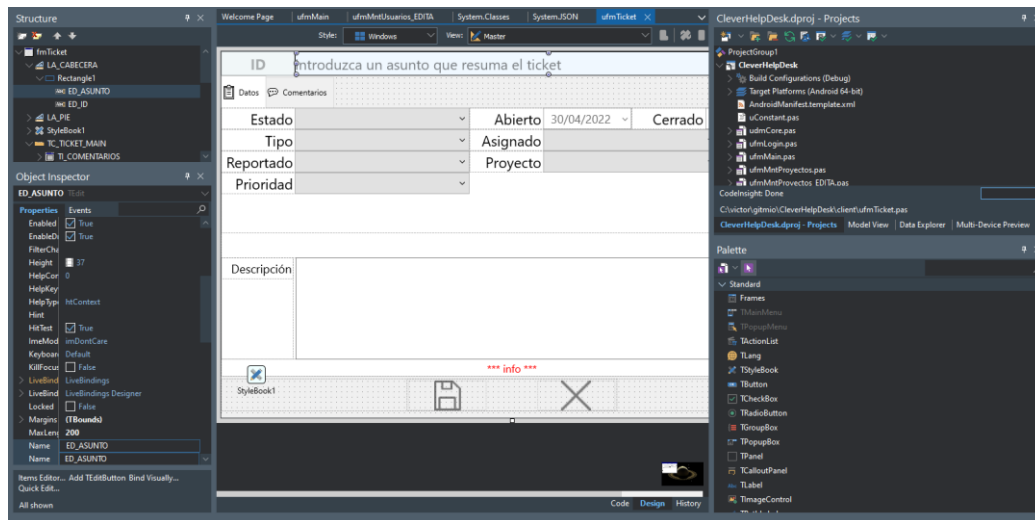


Ilustración 31. Ejemplo diseño formulario ticket [Elaboración propia]

Tampoco he profundizado en los estilos, ya que permite aplicar diferentes estilos (modo oscuro, azul Windows, etc o incluso personalizarlos) pero por falta de tiempo no he podido investigar esta opción.

6.3.3 Compilación multiplataforma

Uno de los puntos fuertes del FIREMONKEY es que con el mismo código puedes hacer una aplicación compilada nativa para las principales plataformas del mercado.

Esto lo podemos definir por proyecto, es decir si es una aplicación de escritorio podríamos indicar que solo se pueda compilar para WINDOWS y MACOS. Configuraremos las plataformas disponibles en el navegador de proyectos y dentro del proyecto en el nodo target platforms.

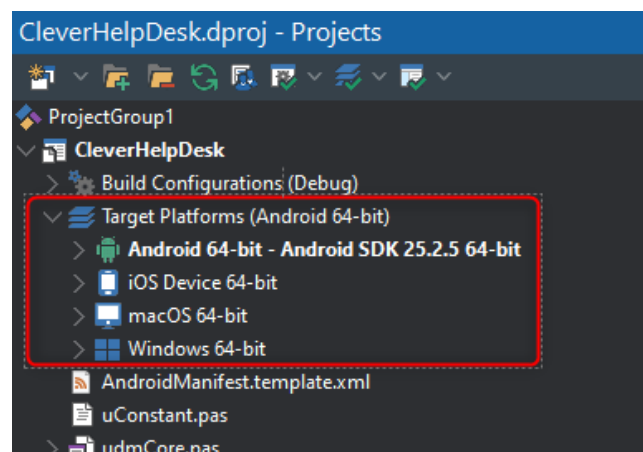


Ilustración 32. Navegador de proyectos del DELPHI [Elaboración propia]

Una vez estamos desarrollando y queremos probar en una plataforma u otra podríamos seleccionar donde queremos que se ejecute:

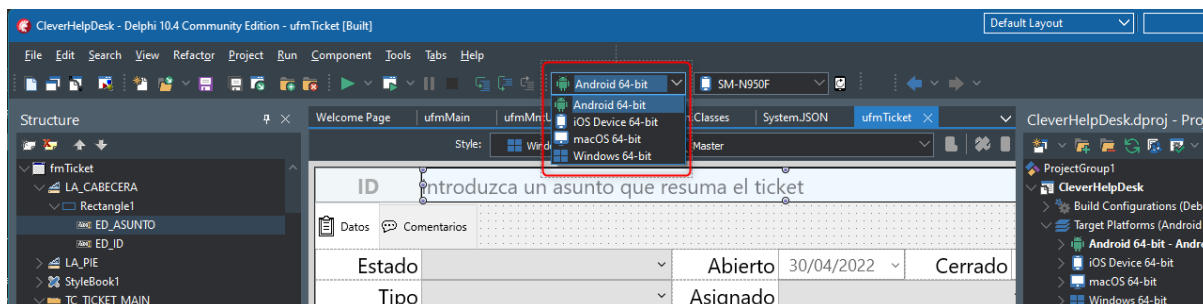


Ilustración 33. Selección plataforma [Elaboración propia]

El resultado será una aplicación WINDOWS o ANDROID, en el caso de IOS y MACOS necesita tener un ordenador de esta marca para hacer la compilación y las pruebas.

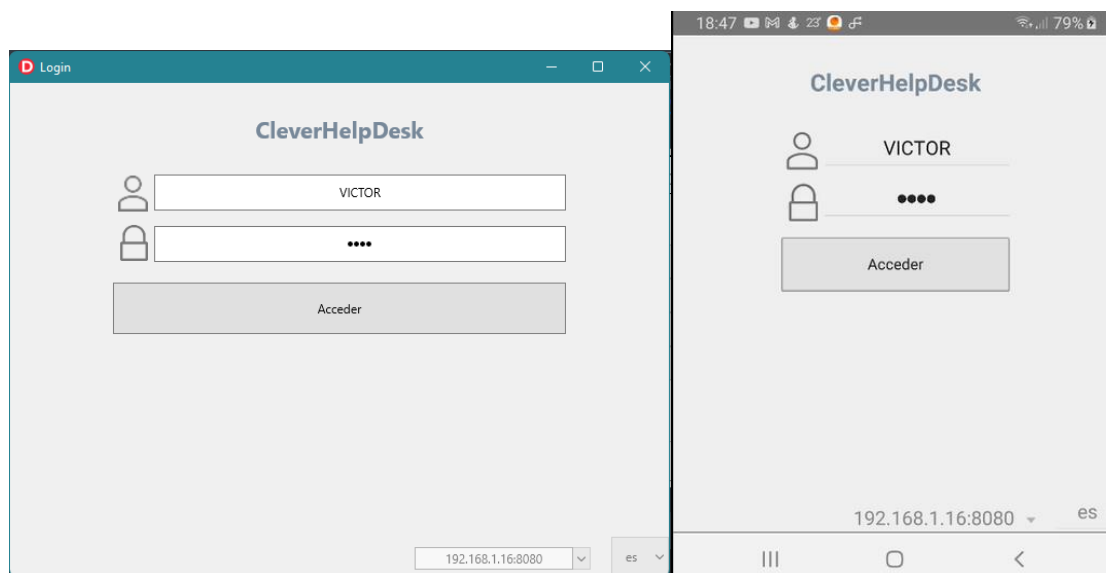


Ilustración 34. Ejemplo pantalla LOGIN aplicación WINDOWS y ANDROID respectivamente [Elaboración propia]

7 Despliegue y pruebas

7.1 Especificaciones y manual despliegue de la aplicación

Requerimientos de la aplicación: para funcionar correctamente la aplicación necesitaría el siguiente software base.

- Instancia de base de datos ORACLE 11XE (es posible que superiores también funcione pero no se ha verificado)
- Servidor con JAVA 11 con WINDOWS (se ha probado con WINDOWS 11, pero no debería haber problema en utilizarlo con WINDOWS SERVER o LINUX)
- Aplicación cliente:
 - Módulo WINDOWS: necesita WINDOWS11 (solo se ha probado con WINDOWS 11 pero seguramente funcionará con versiones anteriores y server)
 - Módulo ANDROID: necesita ANDROID9 (solo se ha probado con ANDROID9, pero en el minSdkManifest se configura con la versión 23, por lo que debería soportar a partir de ANDROID6)

Compilación de los fuentes: los fuentes se pueden descargar desde la carpeta drive compartida o del repositorio GIT público <https://github.com/victorgv/CleverHelpDesk>.

- Servidor, están en la carpeta “server” (raíz_CleverHelpDesk\server) y la compilación se realiza utilizando el IDE *IntelliJ IDEA 2021.2.3 (Community Edition)*.
 - Configuración de la base de datos en el fichero *application.properties*:

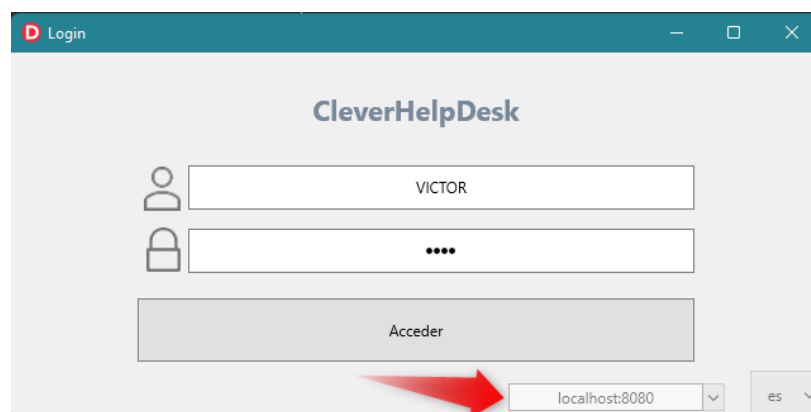
```
# ORACLE -> configuración DB
spring.datasource.url=jdbc:oracle:thin:@192.168.1.9:1521:XE
spring.datasource.username=cleverhelpdesk
spring.datasource.password=12345678
spring.datasource.driver-class-name=oracle.jdbc.OracleDriver
```

- Configuración del proveedor email, también en el fichero *application.properties*:

```
#Email RECEPCION
email.address=cleverhelpdesk.email@gmail.com
email.username=cleverhelpdesk.email%40gmail.com
email.password=${PASSWORD1}
email.server.host=imap.gmail.com
email.server.port=993

#Email ENVÍO
spring.mail.host=smtp.gmail.com
spring.mail.port=587
spring.mail.username=cleverhelpdesk.email@gmail.com
spring.mail.password=${PASSWORD1}
```

- Cliente, están en la carpeta “client” (raíz_CleverHelpDesk\client) y la compilación se realiza utilizando el IDE *Delphi 10.4 Community Edition*.
 - Configuración de la IP/PUERTO API REST: se puede indicar en la unidad ufmLogin, en el componente ED_SERVER, propiedad Text, aunque para facilitar las pruebas también se puede cambiar directamente en la ejecución desde la pantalla de LOGIN:



Despliegue:

- Base de datos: es necesario crear el esquema de base de datos utilizando un usuario SYSTEM con el siguiente script:


```
create user cleverhelpdesk identified by PASSW_ELEGIDO;
grant connect,resource to cleverhelpdesk;
```
- Otros scripts de BD: no se requieren ya que la propia aplicación de servidor crea las tablas y rellena las maestras con los datos necesarios. Creará un usuario “ADMIN” con passw “admin”.
- Aplicación servidor API REST: para arrancar el programa desde la ruta del servidor lanzar el comando Maven: `mvn spring-boot:run`

- Aplicación cliente:
 - Desde WINDOWS: la ruta ..\client\Win64\Debug lanzar el ejecutable CleverHelpDesk.exe
 - Desde ANDROID: en la ruta ..\client\Android64\Debug\CleverHelpDesk\bin copiar e instalar el fichero CleverHelpDesk.apk en el dispositivo.

7.2. Pruebas de caja negra

A continuación se detallan las pruebas que se han realizando para comprobar el correcto funcionamiento de los diferentes casos de uso.

CU_01 LOGIN		
Objetivo probado	Requisitos probados	Pruebas a realizar
LOGIN correcto y obtención TOKEN JWT	Se comprueba capacidad de hacer LOGIN sobre el API devolviendo el TOKEN JWT para que el resto de peticiones lo utilicen	Arrancar la aplicación, introducir un usuario/passw correcto y comprobar que se accede
Error "UNAUTHORIZED" si el TOKEN expira	Por defecto están configurados los TOKENs para que expiren a los 60'	Ver que al expirar se muestra un aviso en pantalla "TOKEN expirado" y se vuelve a pedir LOGIN. Probar desde POSTMAN (no lo permitirá el CLIENTE) que con el TOKEN expirado no se puede atacar al API
LOGIN incorrecto	No se permitirá entrar en la aplicación ni utilizar el API si no hay un LOGIN correcto	Con el cliente intentaremos hacer LOGIN con un usuario incorrecto, tenemos que comprobar que no se puede acceder y que muestra un mensaje apropiado

CU_02 Consultar TICKETS		
Objetivo probado	Requisitos probados	Pruebas a realizar
Consulta tickets asignados a un agente	Que se devuelven los tickets, concretamente los pertenecientes a un agente	Configurar el filtro para que indique un agente y lanzar la consulta verificando que todos los tickets son de ese usuario
Consulta tickets por rango de fechas	Se comprueba que se aplica correctamente el filtro por rango de fechas y no se puede superar el rango máximo que son 366 días	Cambiar el rango de fechas por defecto, lanzar la consulta y verificar que los tickets devueltos corresponden a la fecha indicada. Verificar también que no se pueden lanzar consultas de más de 366 días.
Consulta cuando el perfil es "USUARIO"	Comprobar que solo puede ver los tickets que ha abierto	Utilizando un usuario del tipo usuario comprobaremos que solo verá sus tickets y no los de otros usuarios

CU_03 Creación TICKET (enviando un email)		
Objetivo probado	Requisitos probados	Pruebas a realizar
Creación tickets desde email	1 La recepción del ticket crea un email 2 se comprueba que el usuario (email) es correcto y envía una respuesta	1 Enviar un email a la dirección cleverhelpdesk.email@gmail.com y comprobar que se crea en la aplicación. 2 verificar si encuentra el usuario (por la dirección de email) le responderá con otro email indicando que se ha creado el ticket

CU_03 Creación TICKET (desde el formulario de creación)		
Objetivo probado	Requisitos probados	Pruebas a realizar
Creación tickets desde la aplicación cliente	1 Los usuarios con perfil ADMINISTRADOR y AGENTE puede crear un ticket. 2 Un usuario con perfil "USUARIO" no podrá.	1 Con un usuario que tenga estos perfiles entrar en la aplicación y verificar que tiene activa la opción de crear ticket. Crear un ticket de prueba. 2 con un usuario que no tenga perfil ADMINISTRADOR o AGENTE verificar que la aplicación no muestra la opción de crear tickets
El alta de un ticket nuevo emite un email al "remitente"	Al crear un ticket se emite un email al remitente con el número de ticket.	Crear un ticket y verificar que el remitente recibe el email de alta

CU_04 Gestión del TICKET		
Objetivo probado	Requisitos probados	Pruebas a realizar
Solo puede acceder si se tiene el permiso adecuado	Vamos a verificar que solo los usuarios con perfil ADMINISTRADOR y AGENTE pueden entrar a gestionar el ticket (hacer cambios de estado, asignar a un agente, etc)	Probaremos primero con un usuario que no tenga permiso ADMINISTRADOR o AGENTE, verificaremos que no puede cambiar el estado, asignar un agente, etc. Después probaremos con un usuario ADMINISTRADOR o AGENTE y verificaremos que efectivamente puede hacer cambios
Al cerrar un TICKET se emitirá un email al remitente	Verificaremos que el sistema detecta el cambio de estado y emite el email de cierre para avisar al usuario que ya se ha terminado	Cerraremos un ticket y verificaremos que el sistema emite un email al destinatario.

CU_05 Adjuntar Comentarios al TICKET		
Objetivo probado	Requisitos probados	Pruebas a realizar
Añadir comentarios a un ticket	Si es perfil USUARIO solo podrá añadir comentarios a sus tickets. Perfil ADMINISTRADOR, AGENTE puede añadir comentarios a cualquier ticket	Insertar un comentario y verificar que se graba correctamente. Verificar que no se pueden cargar comentarios vacíos.
Visualización comentarios	Desde el formulario del ticket se podrán ver los comentarios introducidos	Después de grabar el comentario anterior salir y entrar en el ticket para verificar que se muestra en pantalla.

CU_06 Gestión de Usuarios		
Objetivo probado	Requisitos probados	Pruebas a realizar
Permisos	Solo un usuario con permiso ADMINISTRADOR puede acceder a la Gestión Usuarios	Probar con un usuario que no tiene perfil ADMINISTRADOR y verificar que el sistema no permite acceder. Probar de nuevo con un usuario que tenga permisos ADMINISTRADOR y verificar que sí se puede acceder.
Creación/Modificación usuarios	La creación de usuarios nuevos y la modificación de usuarios existentes.	Crear un usuario nuevo. Modificar el usuario creado anteriormente.
Acceso Usuario y seguridad	Verificar que el usuario nuevo puede entrar en el sistema y que el passw se graba en BD encriptado	Hacer login con el usuario nuevo. Inspeccionar la tabla USER_ de base de datos para confirmar que el passw está encriptado.
Validaciones duplicados	Se verificará que no puedan existir dos usuarios con el mismo <i>userName</i> o/y el mismo <i>email</i> .	Un usuario creado previamente no permitirá modificar el <i>userName</i> , en cambio sí permitirá cambiar el <i>email</i> . Probar crear un usuario nuevo duplicando el <i>userName</i> de otro usuario previamente y verificar que el sistema no lo permite.
Validación email	Se verificará que la dirección email introducida tiene un formato correcto	Crear un usuario introduciendo un email con formato incorrecto y verificar que la aplicación lo detecta y alerta

CU_07 Creación Proyectos		
Objetivo probado	Requisitos probados	Pruebas a realizar
Permisos	Solo un usuario con permiso ADMINISTRADOR puede acceder a la pantalla de creación de proyectos.	Probar con un usuario que no tiene perfil ADMINISTRADOR y verificar que el sistema no le permite acceder. Probar de nuevo con un usuario que tenga permisos ADMINISTRADOR y verificar que sí se puede acceder.
Creación/Modificación proyectos	La creación de proyectos nuevos y la modificación de usuarios existentes.	Crear un proyecto nuevo. Modificar los datos del proyecto creado anteriormente.

8 Conclusiones

Mi primer objetivo cumplido para este proyecto era empaparme en todo lo posible en las tecnologías SPRING BOOT y también FIREMONKEY.

SPRING BOOT ha sido todo un acierto, y en el mundo actual y la tendencia a crear “microservicios” es la herramienta perfecta. Voy a seguir formándome en esta tecnología.

La parte de interfaces gráficas con FIREMONKEY le he visto mucho potencial, aunque pensaba que sería más sencillo la curva de aprendizaje es lenta. Pero una vez te acostumbras y entiendes la forma de trabajar el desarrollo se acelera. También la parte de utilizar estilos es espectacular para hacer aplicaciones realmente impactantes, la pena es que en este proyecto no me ha dado tiempo a explorarlo, pero lo tengo apuntado en mi lista de TO-DOs.

Hay un tercer punto que me interesaba tocar y era HIBERNATE, me ha gustado para acelerar el desarrollo de aplicaciones pequeñas. No sé si para grandes aplicaciones con muchas entidades y relaciones puede ser fácil perderse, cuando funciona todo bien es perfecto, pero cuando falla algo es difícil depurarlo.

Ya solo puedo agradecer a todos los profesores del ciclo su dedicación, en mi caso que llevo 20 años trabajando en el mundo del desarrollo me ha servido para actualizarme un poco y para coger fuerza para atreverme a probar nuevas tecnologías.

Muchas gracias!

9 Vías futuras

9.1. Objetivos que se plantearon en la petición inicial y no se alcanzaron:

El único punto que no he podido acometer y que estaba recogido en los requisitos funcionales es: “Guardar un pequeño histórico cronológico del ticket, cuando se creo/cambios estados/etc.” por falta de tiempo no he podido realizarlo, al ser una funcionalidad NO esencial lo puede dejar pendiente para desarrollarlo en futuras versiones.

9.2. Futuras mejoras que se incorporarán en el sistema:

A continuación se exploran las próximas funcionalidades que se añadirán en las próximas versiones, lo primero será arrancar en un entorno productivo, con esta primera versión y gracias a las sugerencias de los usuarios/agentes seguro que llegamos a un sistema muy completo.

- **Gestión de la retención de la información**, es muy importante tratar este tema desde las fases iniciales del proyecto para evitar que en el tiempo crezca descontroladamente el tamaño de las bases de datos. En este caso tendríamos un proceso nocturno que eliminaría registros “obsoletos” y se podría configurar por proyecto y/o tipo tarea, por ejemplo podríamos tener una configuración así:
 - Retención tablas LOGs: 1 año
 - Retención tablas TICKETs y sus relacionadas: 5 años (a partir de la fecha de cierre)
 - Retención ficheros adjuntos: 2 años (a partir de la fecha de cierre)
 - Etc.
- **Posibilidad de adjuntar ficheros**, ya sea imágenes, .PDF, .DOC u cualquier otro fichero interesante para el tratamiento del TICKET. Se establecerá un límite de tamaño fichero máximo 25Mb.
- **Integrar con el LDAP corporativo**, aportando las siguientes ventajas:
 - Simplifica al usuario la utilización de los sistemas ya que con un usuario y password acceden a todos los sistemas (que tienen permiso) de la empresa

- Unifica la política de seguridad, por ejemplo, contraseña de mínimo 10 letras, con mínimo 1 letra mayúscula, mínimo 1 letra numérica, etc. Otro punto importante es la obligación de cambiar el password pasados los 30 días.
- Mejora la seguridad, por ejemplo, el usuario se bloquea después de N intentos fallidos, se caduca si pasan más de 30 días sin utilizar el sistema, etc.
- Cuando un usuario causa baja en la empresa automáticamente se dá de baja en LDAP impidiendo que pueda entrar en las aplicaciones.
- **Posibilidad de obtener informes analíticos**, es muy importante poder medir los datos en los sistemas para tener una visión de lo que está ocurriendo y poder atacar a los problemas o tipos de incidencias más comunes. Entre otros aspectos podremos obtener:
 - Cuantos tickets se abren por tipo, también se podrá detallar por proyecto/tipo. Pudiendo extraer la información por rangos de fechas o mensual.
 - El tiempo medio de resolución de tickets (desde que se abre hasta que se cierra) por tipo, o tipo/proyecto. Pudiendo extraer la información por rangos o por meses para poder ver la evolución de los tiempos.
 - Estadística del agente, para poder obtener la productividad, pudiendo obtener datos como número de tickets cerrados, tiempo medio global y por tipo de ticket. Se podrá extraer la información por rangos o por meses para poder ver la evolución del trabajador.
- **Integración directa desde las aplicaciones corporativas**, es decir que los usuarios puedan abrir la aplicación CleverHelpDesk desde otras aplicaciones de escritorio sin necesidad de logins (la aplicación anterior tendrá su login y le dará un token acceso temporal) para agilizar el uso y consulta.
- **Módulo WEB**, se creará un módulo WEB, solo para la funcionalidad del usuario para que pueda abrir/consultar sus tickets.
- **Comentarios Ocultos para el Usuario y envío por email**, en los comentarios del ticket el agente podrá ocultar comentarios, claramente los que sean para la resolución del ticket o indiquen aspectos técnicos que un usuario no debería ver. Los comentarios accesibles al usuario se podrán enviar por email. Por ejemplo, para poder hacer una consulta al usuario por parte del agente y el usuario podrá responder directamente en el email.
- **Convertir la aplicación servidor en un Microservicio**, se trataría de “DOCKERIZAR” el lado del servidor para poder moverla a un contenedor local (CPD

corporativo) o en la nube. Así como permitir crecer en instancias si el volumen de peticiones lo requiere.

10 Bibliografía/Webgrafía

- <https://www.diagrams.net/> aplicación para modelado de diagramas
- <https://openwebinars.net> - curso de Spring Core
- <https://www.youtube.com/watch?v=iFcDoP6jEeE> Casos de uso y diagramas de casos de uso | | UPV
- <https://www.youtube.com/watch?v=orvAkFFWo5o> Diagrama de casos de uso | | UPV
- <https://www.youtube.com/watch?v=JioEGJlq88> Diagrama de clases | | UPV
- https://docwiki.embarcadero.com/RADStudio/Alexandria/en/Multi-Device_Applications_Index Ayuda DELPHI/FIREMONKEY para desarrollo aplicaciones multiplataforma
- <http://spring.io/> Documentación Spring

Anexo A. Glosario

A continuación se incluye la terminología y siglas utilizadas en la elaboración de esta memoria.

API	Una API o interfaz de programación de aplicaciones es un conjunto de definiciones y protocolos que se usa para diseñar e integrar el software de las aplicaciones.
BBDD	Una base de datos o banco de datos es un conjunto de datos almacenados sistemáticamente para su posterior uso.
DELPHI	Embarcadero Delphi, antes conocido como CodeGear Delphi, Inprise Delphi y Borland Delphi, es un entorno de desarrollo de software diseñado para la programación de propósito general con énfasis en la programación visual.
Diagrama Casos de Uso	Describe el comportamiento del sistema desde el punto de vista de un usuario que interactúa con él.
Diagrama Entidad-Relación	Un diagrama entidad-relación es una herramienta para el modelo de datos, la cual facilita la representación de entidades de una base de datos.
DOCKER	Docker es un sistema operativo para contenedores. De manera similar a cómo una máquina virtual virtualiza (elimina la necesidad de administrar directamente) el hardware del servidor, los contenedores virtualizan el sistema operativo de un servidor.
DTO	(Data Transfer Object). Los DTO son un tipo de objetos que sirven únicamente para transportar datos.
Entidad	Podemos definir las entidades como la representación de aquellos elementos (físicos o abstractos) de los que se desea almacenar la información.
framework	Un framework es un esquema o marco de trabajo que ofrece una estructura base para elaborar un proyecto con objetivos específicos, una especie de plantilla que sirve como punto de partida para la organización y desarrollo de software
Github	GitHub es un sitio "social coding". Te permite subir repositorios de código para almacenarlo en el sistema de control de versiones Git.
Grid	Es una matriz bidimensional donde representar registros de información, ya sea texto, imágenes u otro tipo de componente
Help Desk	Mesa de Ayuda, o Centro de Servicio, o simplemente CAU Centro de Atención al Usuario es un 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 con las Tecnologías de la Información y la Comunicación (TIC).
HIBERNATE	¿Qué es el Hibernate? Resultado de imagen de hibernate Hibernate es una herramienta de mapeo objeto-relacional (ORM) bajo licencia GNU LGPL para Java, que facilita el mapeo de atributos en una base de datos tradicional, y el modelo de objetos de un aplicación

HTTPS	mediante archivos declarativos o anotaciones en los beans de las entidades que permiten establecer estas relaciones HyperText Transfer Protocol Secure
Inversión de control	¿Qué es Inversion of Control y Dependency Injection? En inglés, conocido como Inversion of Control (IoC), es un estilo de programación en el cual un framework o librería controla el flujo de un programa. Esto es un cambio con respecto a paradigmas tradicionales donde el programador especifica todo el flujo del programa.
Ticket	Un ticket es un archivo contenido en el sistema de seguimiento de incidencias que contiene información acerca de intervenciones de software hechas por personal de soporte técnico o terceros a pedido de un usuario final que ha reportado un incidente que está impidiéndoles trabajar en sus computadoras cuando ellos esperaban poder hacerlo. Los tickets se crean generalmente en un ambiente de help desk o call center. Típicamente el ticket tiene un número único de referencia, también conocido como un número de caso, incidente o reporte de llamada, el cual es usado para permitir al cliente o al personal de soporte localizar, añadir o comunicar el estado del incidente o requerimiento.

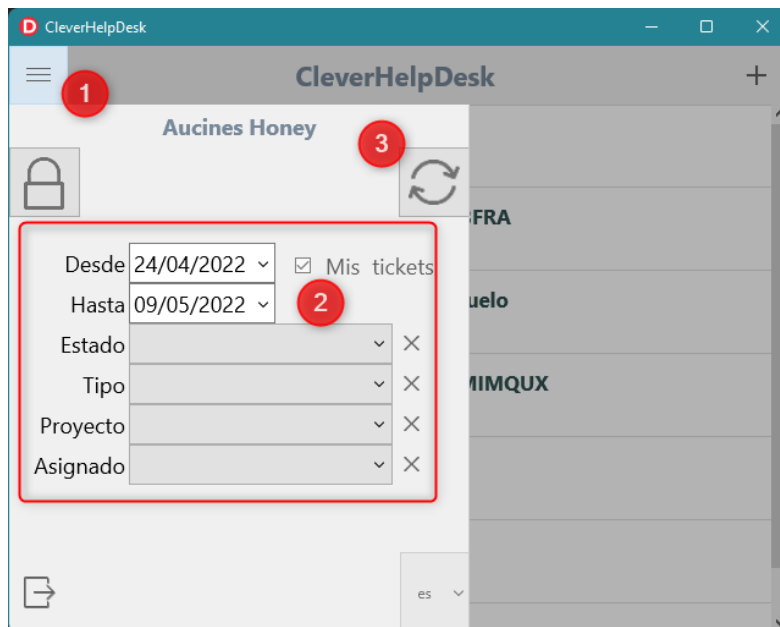
Anexo B. Manual usuario

Login:

El usuario debe estar registrado, para entrar debe introducir el código de usuario y contraseña.

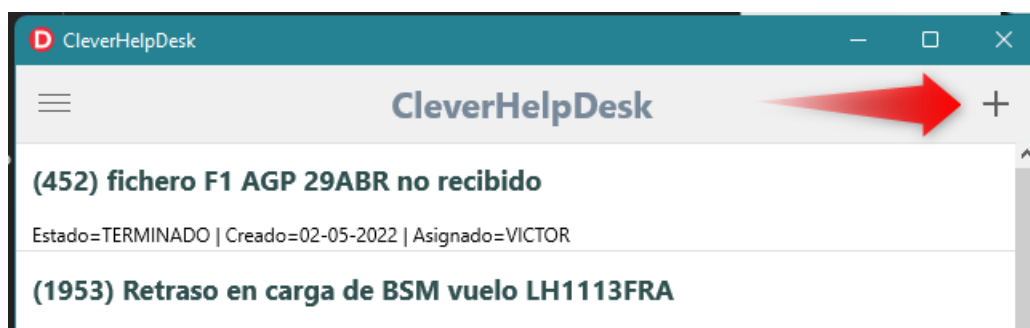
Filtrado de Tickets:

Para buscar y localizar tickets utilizaremos la consulta, accederemos pulsando el botón (1), rellenaremos los filtros deseados (2) y pulsaremos el botón refrescar (3)



Creación de tickets:

Para crear un ticket nuevo pulsaremos en este botón:



Completamos la información del ticket y pulsaremos grabar:

Creación Nuevo Ticket

Incidencia en la recepción de télex BSM

Datos

Comentarios

Estado

Registrado

▼

Abierto

09/05/2022

▼

Cerrado

__/__/__

▼

Tipo

Cahída/error sistema externo

▼

Asignado

▼

Reportado

Aucines Honey

▼

Proyecto

▼

Prioridad

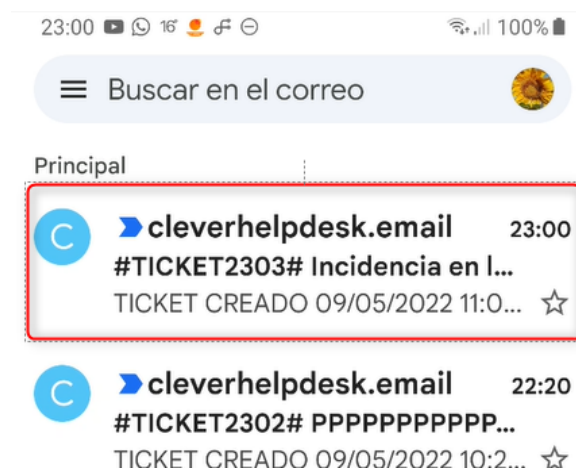
Normal

▼

Descripción

No llegan los mensajes

Se recibirá un email de confirmación al grabar el ticket:

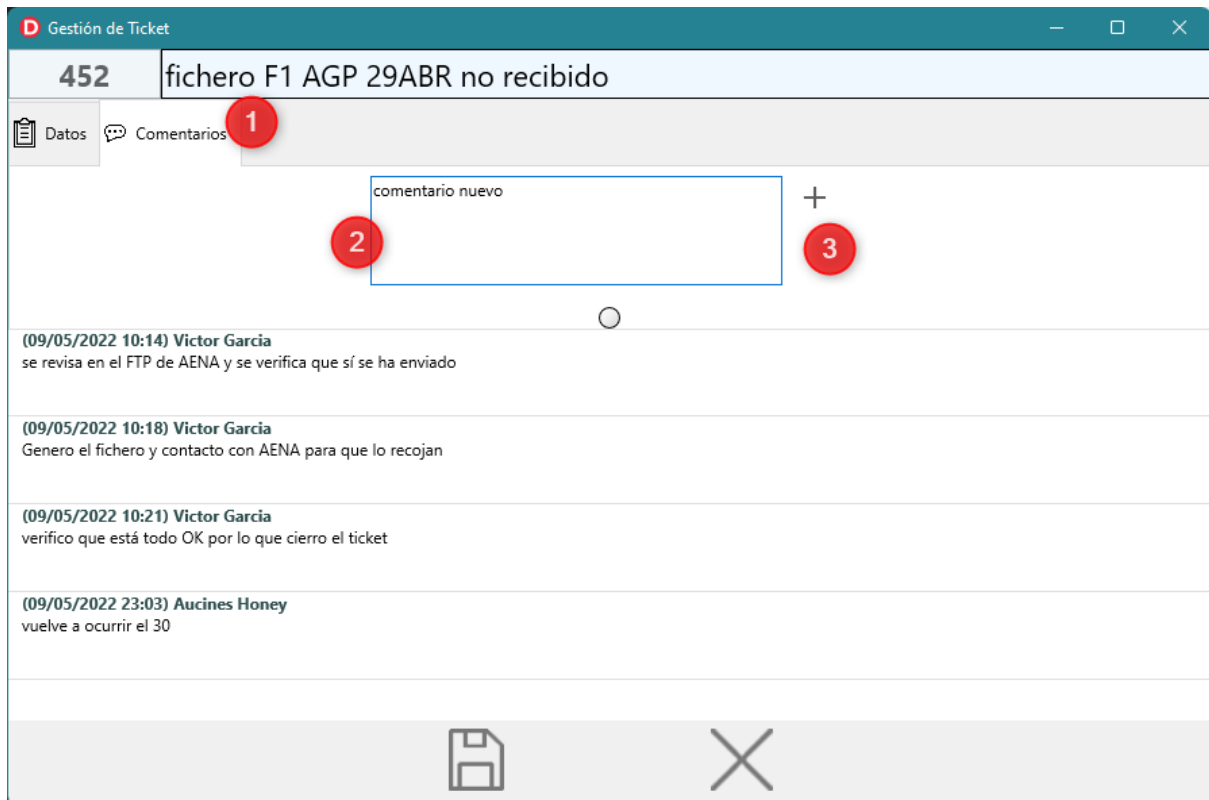


Creación de ticket enviando un email:

Se puede crear un ticket enviando directamente un email a la dirección electrónica del sistema: cleverhelpdesk.emal@gmail.com

Añadir comentarios al ticket:

En cualquier momento se pueden añadir comentarios al ticket, lo realizaremos desde la pestaña comentarios (1), añadimos el texto nuevo (2) y pulsamos añadir (3).



Gestión de Ticket

452 fichero F1 AGP 29ABR no recibido

Datos Comentarios **1**



comentario nuevo **2** **3**

(09/05/2022 10:14) Victor Garcia
se revisa en el FTP de AENA y se verifica que sí se ha enviado

(09/05/2022 10:18) Victor Garcia
Genero el fichero y contacto con AENA para que lo recojan

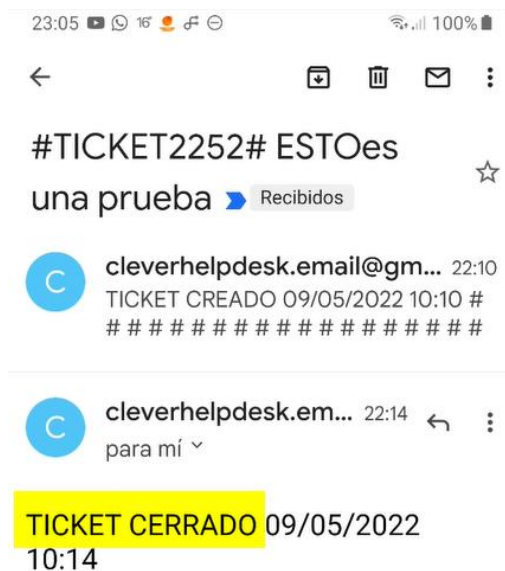
(09/05/2022 10:21) Victor Garcia
verifico que está todo OK por lo que cierro el ticket

(09/05/2022 23:03) Aucines Honey
vuelve a ocurrir el 30

Email cierre ticket:

Una vez se cierre el ticket el usuario recibira un email.



Anexo C. Capturas de Pantalla

Utilización del Postman:

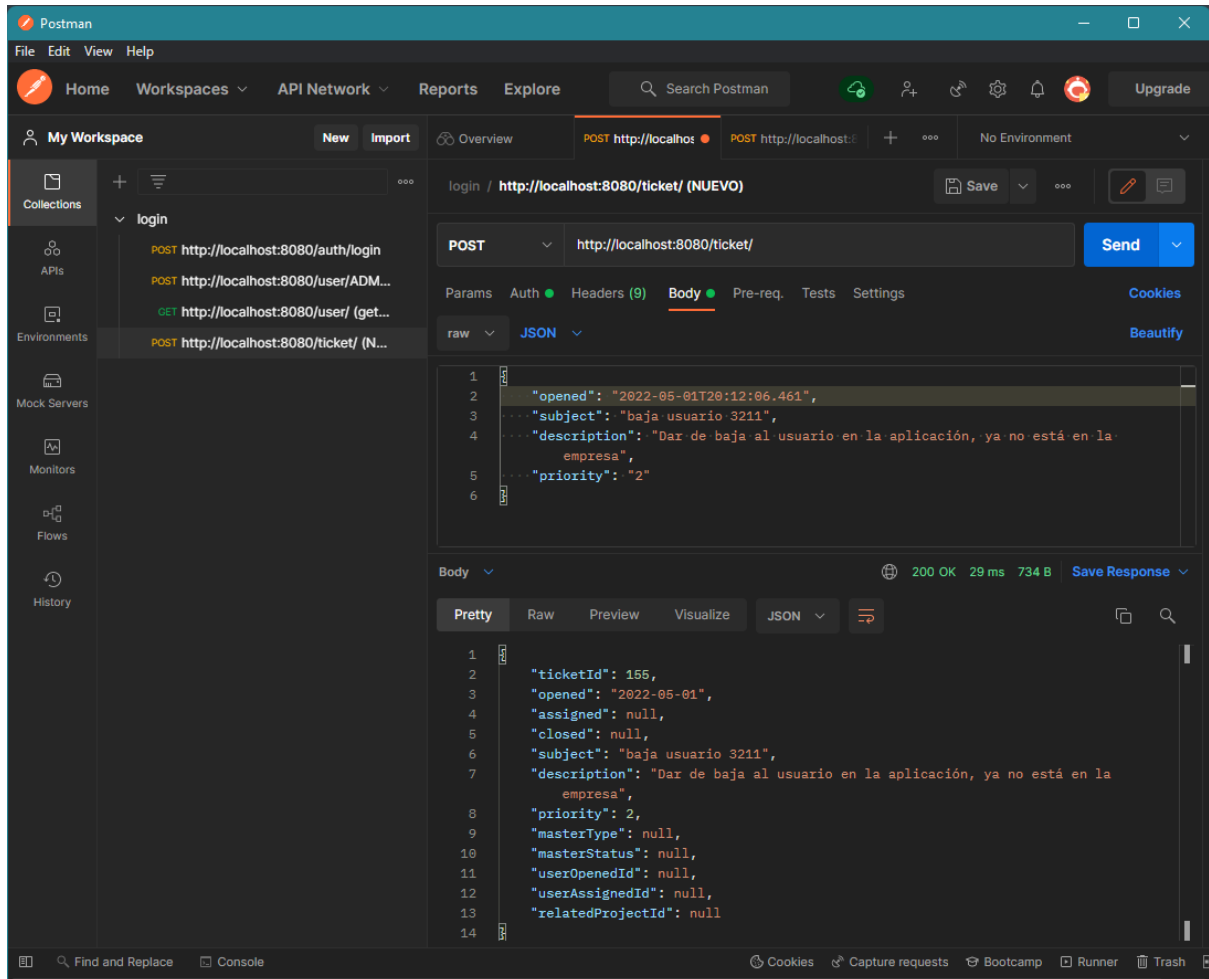


Ilustración 35. Creación de un ticket desde postman [Elaboración propia]

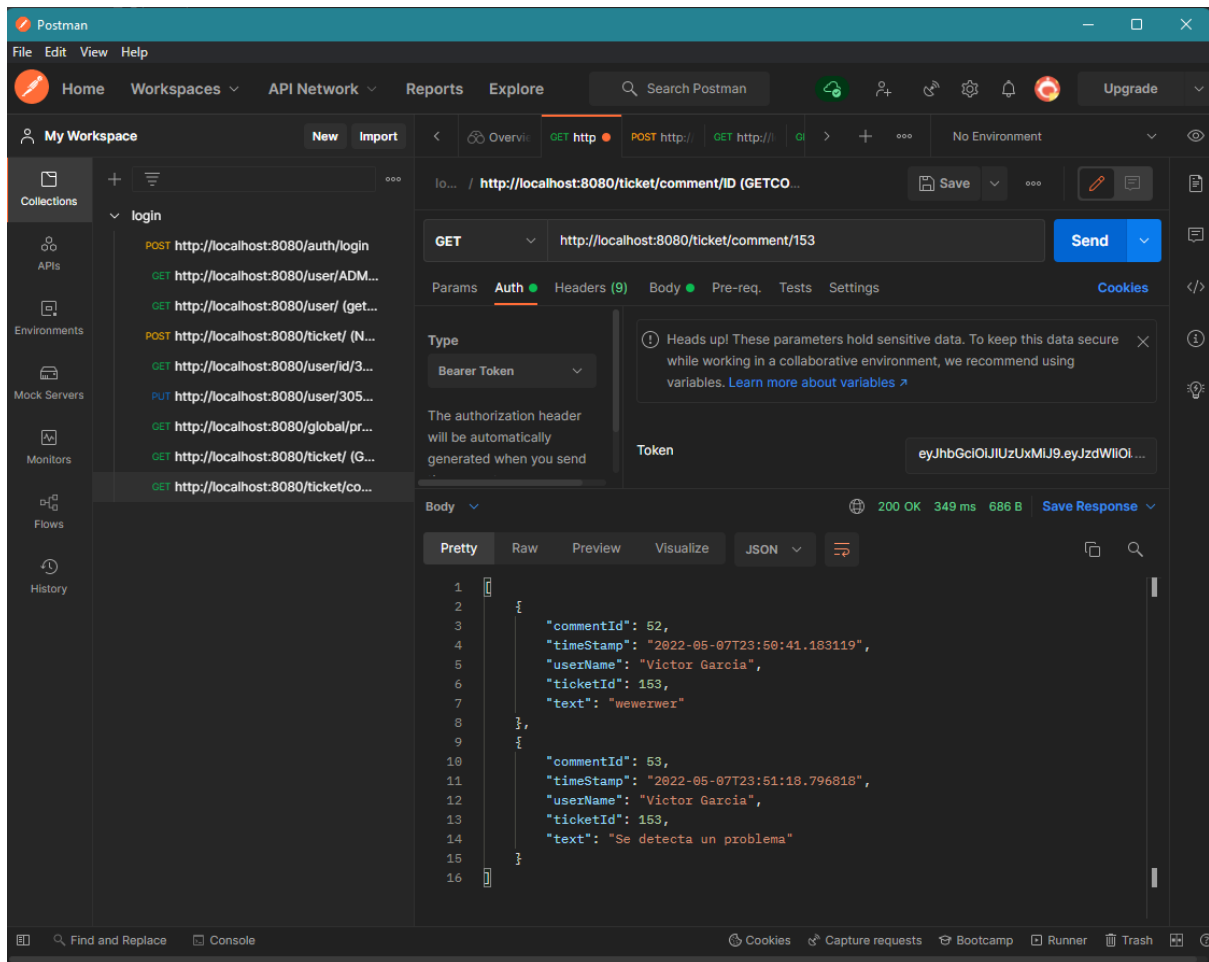


Ilustración 36. Listado de los comentarios del ticket con ID 153 [Elaboración propia]

Anexo D. Gantt final del proyecto

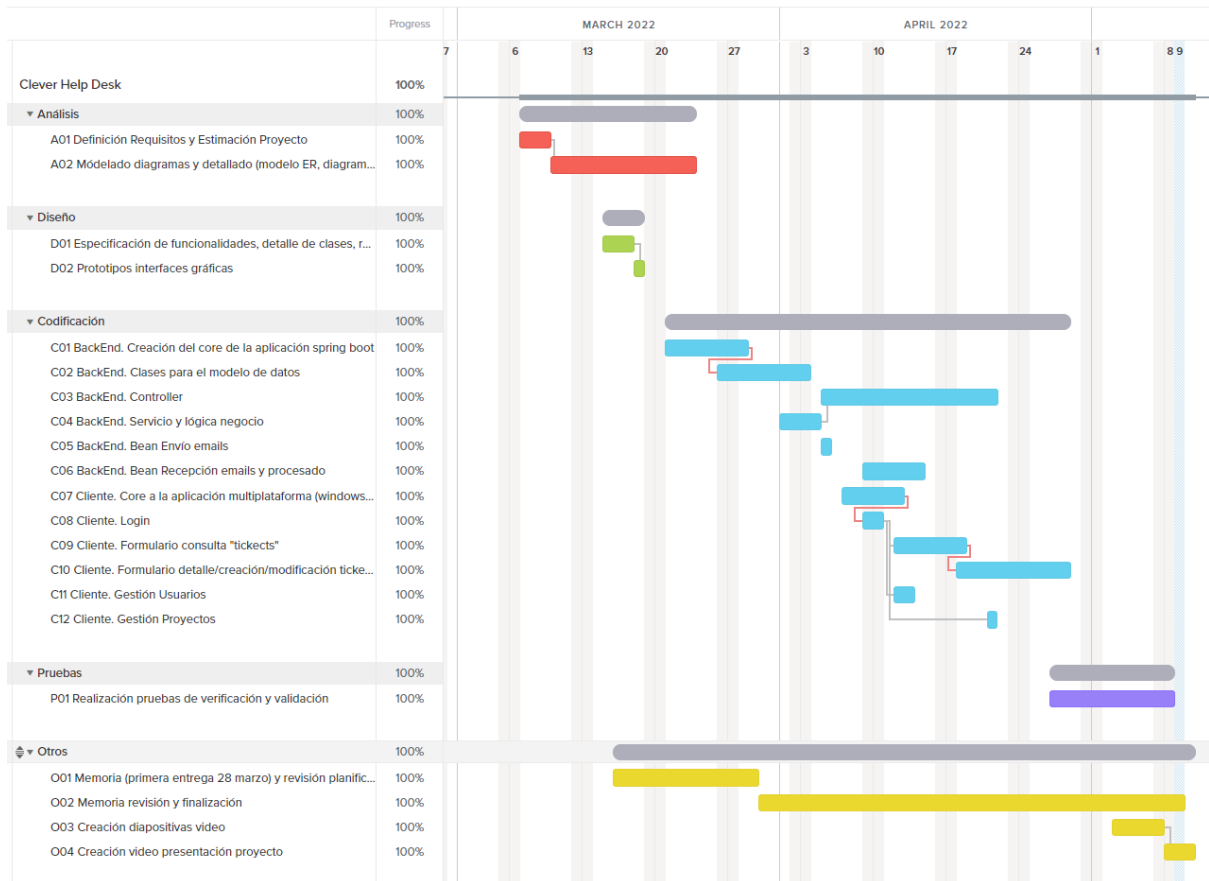


Ilustración 37. Gantt actualizado al finalizar el proyecto [Elaboración propia utilizando teamgantt.com]