Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Mención Ingeniería del Software

Automatización para entornos ITSM-ITIL basada en bots

Autor:

D. Juan Carlos Gil Díaz

Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Mención Ingeniería del Software

Automatización para entornos ITSM-ITIL basada en bots

Autor:

D. Juan Carlos Gil Díaz

Tutor interno:

D. Diego Rafael Llanos Ferraris

Tutor externo:

D. Francisco Vallejo Luna

1- Resumen

Bot de Discord que se comunica con la aplicación Service Manager a través de su API REST y que permite a los usuarios la creación, modificación, comprobación de estado y cierre de Incidencias, además de la consulta de diversos Indicadores clave de rendimiento, todo ello a través del chat de Discord.

Palabras clave: Bot, Discord, Service Manager, REST.

1.1- Abstract

Discord Bot which communicates with Service Manager application using its REST API, which allows users to create, modify, check the status of and close an Incident, in addition to the consult of various Key Performance Indicators, all done though Discord's chat.

Key words: Bot, Discord, Service Manager, REST.

2- Tabla de contenidos

3- Lista de figuras y tablas

4- Introducción

4.1- Objetivo

El objetivo del proyecto ha sido el de automatizar la creación de Incidencias en la aplicación Service Manager mediante el desarrollo de un bot de Discord que interactúe con los usuarios (operadores) sin necesidad de que estos utilicen directamente la aplicación y de manera rápida y fácil mediante el uso del API REST de Service Manager. Los casos de uso a implementar son los siguientes:

- Creación de nuevas incidencias en el sistema.
- Consulta del estado de una incidencia.
- Modificación de datos de una o varias incidencias.
- Cierre de una incidencia.
- Consulta de uno o varios Indicadores Clave de Rendimiento (KPI)

4.2- Motivación

La motivación para el desarrollo del proyecto se debe a la **curva de aprendizaje** presentada por Service Manager, que provoca que su rendimiento sea bajo durante el comienzo del uso de la aplicación, ya que desde un principio se presentan muchos campos para cada registro, lo que puede provocar un exceso de información debido a que el usuario no sabe el propósito de todos los campos, o qué campos son necesarios para realizar una determinada acción tal como cambiar el estado de una incidencia creada. El Bot de Discord **simplificará los pasos** en la realización de diversas acciones proporcionando un proceso sencillo y guiado que permitirá al usuario realizar ciertas tareas de forma más sencilla, sin necesidad de utilizar ninguno de los clientes de Service Manager, y manteniendo la seguridad y la integridad de los datos.

4.3- Definiciones

A continuación se definen varios conceptos que se utilizarán a lo largo de este documento:

- **Service Manager (SM)**: Solución desarrollada por Micro Focus que cumple con el estándar ITIL y que permite la gestión de servicios, activos, conocimiento y servicios empresariales. Proporciona una plataforma en la que estandarizar y automatizar procesos, flujos de trabajo y tareas de manera “code-less”.

- **Cliente Windows de SM**: Aplicación cliente de Service Manager que se instala en un dispositivo Windows y que permite el acceso a toda la funcionalidad de Service Manager excepto al creador de flujos de trabajo. Es instalado por el cliente.

- **Cliente Web de SM**: Aplicación cliente de Service Manager que se despliega en un servidor Tomcat y que permite el acceso a toda la funcionalidad de Service Manager exceptuando el diseñador de formularios. Se despliega en el servidor.

- **Operador**: Usuario de Service Manager capaz de iniciar sesión en la aplicación y realizar distintas tareas dependiendo de los permisos que tenga. Debe tener un contacto asociado.

- **Contacto**: Usuario de Service Manager que no puede iniciar sesión en la aplicación. Se utilizan como base para los operadores y como registros a los que podemos asignar “tickets”.

- **Ticket**: Registro de Incidencia, Problema, Petición de cambio o Interacción. Cada ticket posee un flujo de trabajo individual.

- **Incidencia**: Registro de un fallo o problema con algún elemento que previamente funcionaba correctamente. Pueden ser creadas directamente o creadas a partir de una interacción.

- **Problema**: Incidencia que se repite a menudo o que provoca graves daños. Durante la resolución de problemas se comienza con una solución temporal (workaround) mientras se investiga la causa principal del problema (root cause analysis). Finalmente, una vez que se ha encontrado la solución al problema, se crea una petición de cambio y se resuelve el problema.

- **Petición de cambio**: petición para realizar un cambio. Puede ser para corregir un problema conocido o para modificar el funcionamiento de algún elemento registrado. Pasan por una serie de aprobaciones a lo largo de su flujo de trabajo y deben tener al menos un plan de recuperación en caso de fallo durante la aplicación del cambio.

- **Interacción**: Registro del primer contacto del cliente con el personal de asistencia. La interacción puede o bien resolverse de manera directa en ese estado, o bien ser escalada a Incidencia o incluso a Problema dependiendo de su complejidad, del tiempo que lleve abierta, o del conocimiento de los expertos asignados a la incidencia.

- **KPI**: *Key Performance Indicator*, Indicador de Rendimiento Clave. Valores numéricos útiles para medir aspectos clave de rendimiento de los trabajadores o del sistema, tales como media de días entre la apertura y el cierre de un ticket, o el porcentaje de tickets que son escalados.

- **Discord**: Plataforma de mensajería instantánea mediante chat, voz o vídeo lanzada en el año 2015, que permite la creación de servidores privados con canales, además de la integración con bots desarrollados en múltiples lenguajes de programación como Python o Javascript.

- **Bot**: Aplicación diseñada para realizar tareas de manera automática a partir de comandos. En nuestro caso desarrollaremos un bot para Discord que interactuará con usuarios y se comunicará con Service Manager mediante su API REST.

- **Servidor de Discord**: Grupo privado de Discord que contiene uno o más canales mediante los que los usuarios pertenecientes al servidor pueden comunicarse.

- **Canal de Discord**: Chats dentro de servidores. Los canales pueden ser públicos (visibles para todos los usuarios en el servidor) o privados (solo visibles para un determinado número de usuarios). Cada canal tiene su propio chat independiente del resto de canales. Existen canales de texto y de voz, y en los canales de voz se permite la compartición de pantalla con otros usuarios en el canal.

- **Canal privado**: Canal dirigido solo a un determinado número de usuarios en un servidor, no visible para el resto. Pueden ser de voz o de texto.

- **CentOs8**: Sistema operativo de código abierto basado en Linux. Es una bifurcación de GNU/Linux Red Hat Enterprise Linux RHEL. Será descontinuado a finales del año 2021.

- **Micro Focus**: Empresa multinacional de origen británico fundada en el año 1976 dedicada al ámbito de las tecnologías de la Información y el software.

- **ITIL**: *Infrastructure Technologies Information Library*: Conjunto de consejos y buenas prácticas para la gestión y el desarrollo de servicios de Tecnologías de la Información. No es un estándar, puesto que no indica cómo se deben hacer las cosas, si no que indica qué se debe hacer.

- **CI (Configuration Item)**: Todo elemento hardware, software o incluso de personal que posee una empresa.

5- Análisis de requisitos de usuario

5.1- Requisitos del sistema

A continuación se expone la lista de requisitos funcionales y no funcionales que deberá cumplir el sistema desarrollado:

1. RF001 (**Crear Incidencia**): El sistema deberá permitir la creación de incidencias en Service Manager.
2. RF002 (**Consultar Incidencia**): El sistema deberá permitir a un usuario obtener los datos fundamentales de una consulta a partir de su identificador. Se entienden como datos fundamentales aquellos que sean útiles para el operador que realiza la consulta: el identificador de la incidencia, su título, su descripción, sus valores de impacto y gravedad y (si tiene) su solución y el código con el que se cerró la Incidencia.
3. RF003 (**Cerrar Incidencia**): El sistema deberá permitir el cierre de incidencias en Service Manager, aportando una solución y un código de cierre.
4. RF004 (**Modificar Incidencia**): El sistema deberá permitir la modificación de los datos de incidencias no cerradas en Service Manager.
5. RF005 (**Datos mínimos**): El sistema deberá solicitar los mínimos datos necesarios para llevar a cabo los casos de uso, manteniendo la integridad de la base de datos y evitando errores debido a falta de valores.
6. RF006 (**Proceso guiado**): El sistema deberá ir solicitando los mínimos datos necesarios para realizar el caso de uso de manera secuencial y guiada para el usuario. El cumplimiento de este requisito nos proporcionará procesos con una curva de aprendizaje casi nula y sin la sobrecarga de información que da Service Manager a nuevos usuarios.
7. RF007 (**Validación de errores**): El sistema debe validar los datos introducidos por el usuario tan pronto como sea posible, impidiendo la transición a la siguiente etapa mientras el dato introducido no sea válido. Gracias a este requisito evitamos malgastar tiempo del usuario, ya que el caso de uso no podrá continuar hasta que no se haya garantizado la validez del dato introducido. De esta manera evitaremos tanto fallos durante la comunicación con Service Manager como posibles inconsistencias de los datos enviados.
8. RF008 (**Indicación de ID de incidencia importante**): Una vez que se haya realizado la creación de la incidencia, se deberá informar al usuario del identificador de la incidencia, que será utilizado para otros casos de uso. Dado que el identificador será el primer dato solicitado en casi todos los demás casos de uso, este requisito garantiza que el identificador de la nueva incidencia sea devuelto al usuario, destacando además dicho valor para reducir las posibilidades de que el usuario lo pierda o lo ignore.
9. RF009 (**Enlaces a los canales**): El sistema proporcionará un enlace a los canales creados durante la realización del caso de uso, así como un enlace de vuelta al canal general al finalizar el caso de uso. Gracias a este requisito el usuario no tendrá que buscar entre la gran cantidad de canales que se pueden ir creando a lo largo del uso de la aplicación, sino que al iniciar el caso de uso obtendrá el enlace al canal en el que podrá continuar con el caso de uso. De igual manera, el sistema avisa de que ya no atenderá un determinado

canal al finalizar el caso de uso, proporcionando un enlace de vuelta al canal general en el que el usuario podrá iniciar un nuevo caso de uso.

10. RF010 (**Posibilidad de cancelar**): En todo momento del proceso se podrá cancelar el caso de uso en que se encuentre. Este requisito permite al usuario cancelar la creación de la incidencia cuando así lo desee, dejando sin efecto todo el caso de uso y no modificando ningún dato.
11. RF011 (**Canal privado**): Los canales creados para la solicitud de datos para la incidencia no deben ser accesibles para usuarios diferentes al bot y al iniciador del caso de uso. Como la información introducida por el usuario puede ser sensible, los canales no deberán ser accesibles para cualquier otra persona. De esta manera garantizamos la confidencialidad de los datos.
12. RF012 (**Consultar KPI**): El sistema deberá permitir la consulta de uno o varios KPIs almacenados en Service Manager.
13. RNF001 (**Python**): El sistema deberá estar implementado en Python, evitando funciones excesivamente complejas, con métodos robustos y comentarios en el código. Evitando funciones de alta complejidad facilitamos la depuración del código y su futura extensibilidad.
14. RNF002 (**REST**): La comunicación con Service Manager deberá realizarse mediante su API REST. No se podrá utilizar SOAP para la comunicación con Service Manager, y deberán exponerse los servicios REST necesarios desde el lado del servidor, respetando en la medida de lo posible la sintaxis http.
15. RNF003 (**Inglés**): La comunicación con el sistema se realizará en inglés. Aunque la entrada del usuario puede estar en cualquier idioma, toda la comunicación que realizará el bot será en inglés.

5.2- Planificación temporal

Para las estimaciones tanto de tiempo como de coste se ha utilizado la herramienta Microsoft Project, cuya licencia es proporcionada por la Escuela. Para obtener una estimación de la fecha de fin de las tareas más precisa hemos creado un calendario laboral de 4 horas diarias y 7 días a la semana (de 16 a 20h), sin festivos.

La duración estimada de las tareas se ha medido en días (tal y como hemos mencionado previamente, un día son 4 horas de trabajo). Como fecha final del proyecto se ha establecido el día 1 de Junio de 2021, y se han planificado las tareas de forma automática dándoles como inicio su comienzo más tardío. El resultado final es una duración estimada de 177,97 días (aproximadamente unas 712 horas), desglosadas en las tareas que se pueden ver en las siguientes imágenes. Aunque en las siguientes imágenes no aparece, tomar como nombre de las columnas el mostrado en la primera imagen.

Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
Proyecto	**177,97 días**	**mar 19/01/21**	**mar 01/06/21**		
Crear planificación inicial	1,5 días	mar 19/01/21	mié 20/01/21		Gestor de proyecto
Configuracion previa	**14,5 días**	**mié 20/01/21**	**dom 31/01/2**	**2**	
Instalar y configurar máquina virtual	3 días	mié 20/01/21	vie 22/01/21		Tecnico
Instalar y configurar servidor Service Manager	7 días	vie 22/01/21	mié 27/01/21	4	Tecnico
Instalar y configurar cliente Windows de Service Manager	1,5 días	mié 27/01/21	vie 29/01/21	5	Tecnico
Desplegar Cliente Web de Service Manager	1,5 días	mié 27/01/21	vie 29/01/21	5	Tecnico
Instalar certificados y configurar SSL	3 días	vie 29/01/21	dom 31/01/21	7;6	Tecnico

Figura 1 - Estimación de tiempo total y sección configuración previa

CU Crear Incidencia	**22,17 días**	**dom 31/01/2**	**mar 16/02/21**	**3**	
Revisar planificación	1 día	dom 31/01/21	lun 01/02/21		Gestor de proyecto
Análisis	**1 día**	**lun 01/02/21**	**lun 01/02/21**	**10**	
Diagrama CU	1 día	lun 01/02/21	lun 01/02/21		Analista
Requisitos	1 día	lun 01/02/21	lun 01/02/21		Analista
Diseño	**3,5 días**	**lun 01/02/21**	**jue 04/02/21**	**11**	
Diagrama Arquitectura	1 día	lun 01/02/21	mar 02/02/21		Analista
Modelo Dominio	1,5 días	lun 01/02/21	mar 02/02/21		Analista
Diagrama Interacción	2 días	mar 02/02/21	jue 04/02/21	15;16	Analista
Diagrama REST	1,5 días	lun 01/02/21	mar 02/02/21		Analista
Implementación	**7 días**	**mar 02/02/21**	**dom 07/02/2**	**15**	
Configuracion REST SM	1 día	mar 02/02/21	mié 03/02/21		Tecnico
Implementación del CU en el bot	6 días	mié 03/02/21	dom 07/02/21	20	Tecnico
Pruebas	**12,17 días**	**dom 07/02/2**	**mar 16/02/21**	**19**	
Pruebas y depuración	7 días	dom 07/02/21	sáb 13/02/21		Tecnico
Reunión de revisión de progreso	0,5 días	sáb 13/02/21	mar 16/02/21	23	Tecnico;Analista;Gestor

Figura 2 - Estimación de tiempo CU Crear Incidencia

CU Modificar Incidencia	**24,5 días**	**mar 16/02/21**	**dom 07/03/2**	**9**	
Revisar planificacion	0,5 días	mar 16/02/21	mié 17/02/21		Gestor de proyecto
Análisis	**0,5 días**	**mar 16/02/21**	**mié 17/02/21**		**25**
Diagrama CU	0,5 días	mar 16/02/21	mié 17/02/21		Analista
Requisitos	0,5 días	mar 16/02/21	mié 17/02/21		Analista
Diseño	**4,83 días**	**mié 17/02/21**	**sáb 20/02/21**	**27**	
Diagrama Arquitectura	0,5 días	mié 17/02/21	mié 17/02/21		Analista
Modelo Dominio	0,5 días	mié 17/02/21	mié 17/02/21		Analista
Diagrama Interacción	1,5 días	mié 17/02/21	sáb 20/02/21		Analista
Diagrama REST	1 día	mié 17/02/21	jue 18/02/21		Analista
Implementacion	**7 días**	**sáb 20/02/21**	**vie 26/02/21**	**30**	
Configuracion REST SM	1 día	sáb 20/02/21	dom 21/02/21		Tecnico
Implementacion CU en el bot	6 días	dom 21/02/21	vie 26/02/21	36	Tecnico
Pruebas	**12,17 días**	**vie 26/02/21**	**dom 07/03/2**	**35**	
Pruebas y depuración	7 días	vie 26/02/21	mié 03/03/21		Tecnico
Reunión de revisión de progreso	0,5 días	mié 03/03/21	dom 07/03/21	39	Tecnico;Analista;Gestor

Figura 3 - Estimación de tiempo CU Modificar Incidencia

CU Consultar estado de Incidencia	**24,33 días**	**dom 07/03/2**	**jue 25/03/21**	**25**	
Revisar planificacion	0,5 días	dom 07/03/21	dom 07/03/21		Gestor de proyecto
Análisis	**0,5 días**	**dom 07/03/2**	**lun 08/03/21**	**42**	
Diagrama CU	0,5 días	dom 07/03/21	lun 08/03/21		Analista
Requisitos	0,5 días	dom 07/03/21	lun 08/03/21		Analista
Diseño	**4,17 días**	**lun 08/03/21**	**jue 11/03/21**	**43**	
Diagrama Arquitectura	0,5 días	lun 08/03/21	lun 08/03/21		Analista
Modelo Dominio	0,5 días	lun 08/03/21	lun 08/03/21		Analista
Diagrama Interacción	1,5 días	lun 08/03/21	jue 11/03/21		Analista
Diagrama REST	1 día	lun 08/03/21	lun 08/03/21		Analista
Implementacion	**7 días**	**jue 11/03/21**	**mar 16/03/21**	**46**	
Configuracion REST SM	1 día	jue 11/03/21	jue 11/03/21		Tecnico
Implementacion CU en el bot	6 días	jue 11/03/21	mar 16/03/21	52	Tecnico
Pruebas	**12,17 días**	**mar 16/03/21**	**jue 25/03/21**	**51**	
Pruebas y depuración	7 días	mar 16/03/21	dom 21/03/21		Tecnico
Reunión de revisión de progreso	0,5 días	dom 21/03/21	jue 25/03/21	55	Tecnico;Analista;Gestor

Figura 4 - Estimación de tiempo CU Consultar estado de Incidencia

CU Cerrar Incidencia	**24,33 días**	**jue 25/03/21**	**lun 12/04/21**	**41**	
Revisar planificacion	0,5 días	jue 25/03/21	jue 25/03/21		Gestor de proyecto
Análisis	**0,5 días**	**jue 25/03/21**	**vie 26/03/21**	**58**	
Diagrama CU	0,5 días	jue 25/03/21	vie 26/03/21		Analista
Requisitos	0,5 días	jue 25/03/21	vie 26/03/21		Analista
Diseño	**4,17 días**	**vie 26/03/21**	**lun 29/03/21**	**59**	
Diagrama Arquitectura	0,5 días	vie 26/03/21	vie 26/03/21		Analista
Modelo Dominio	0,5 días	vie 26/03/21	vie 26/03/21		Analista
Diagrama Interacción	1,5 días	vie 26/03/21	lun 29/03/21		Analista
Diagrama REST	1 día	vie 26/03/21	sáb 27/03/21		
Implementacion	**7 días**	**lun 29/03/21**	**sáb 03/04/21**	**62**	
Configuracion REST SM	1 día	lun 29/03/21	mar 30/03/21		Tecnico
Implementacion CU en el bot	6 días	mar 30/03/21	sáb 03/04/21	68	Tecnico
Pruebas	**12,17 días**	**sáb 03/04/21**	**lun 12/04/21**	**67**	
Pruebas y depuración	7 días	sáb 03/04/21	jue 08/04/21		
Reunión de revisión de progreso	0,5 días	jue 08/04/21	lun 12/04/21	71	Tecnico;Analista;Gestor

Figura 5 - Estimación de tiempo CU Cerrar Incidencia

CU Consultar KPI	**36,63 días**	**lun 12/04/21**	**lun 10/05/21**	**57**	
Revisar planificacion	2 horas	lun 12/04/21	mar 13/04/21		Gestor de proyecto
Análisis	**4,47 días**	**mar 13/04/21**	**vie 16/04/21**	**74**	
Diagrama CU	0,5 días	mar 13/04/21	mar 13/04/21		Analista
Requisitos	0,8 días	mar 13/04/21	vie 16/04/21		Analista
Diseño	**2 días**	**vie 16/04/21**	**dom 18/04/2**	**75**	
Diagrama Arquitectura	0,5 días	vie 16/04/21	vie 16/04/21		Analista
Modelo Dominio	1 día	vie 16/04/21	sáb 17/04/21		Analista
Diagrama Interacción	1 día	vie 16/04/21	sáb 17/04/21		Analista
Diagrama REST	2 días	vie 16/04/21	dom 18/04/21		Analista
Implementacion	**14 días**	**dom 18/04/2**	**mié 28/04/21**	**78**	
Configuracion REST SM	2 días	dom 18/04/21	lun 19/04/21		Tecnico
Creación de scripts SM	7 días	lun 19/04/21	sáb 24/04/21	84	Tecnico
Implementacion CU en el bot	5 días	sáb 24/04/21	mié 28/04/21	85	Tecnico
Pruebas	**15,5 días**	**mié 28/04/21**	**lun 10/05/21**	**83**	
Pruebas y depuración	7 días	mié 28/04/21	lun 03/05/21		Tecnico
Reunión de revisión de progreso	0,5 días	lun 03/05/21	lun 10/05/21	88	Tecnico;Analista;Gestor

Figura 6 - Estimación de tiempo CU Consultar KPI

Revisión final de calidad del código	4 días	lun 10/05/21	jue 13/05/21	73	Tecnico
Crear Documento Proyecto	**30 días**	**lun 10/05/21**	**mar 01/06/21**	**73**	**Tecnico**
Crear documento del proyecto	30 días	lun 10/05/21	mar 01/06/21		Tecnico
Revisar y exportar diagramas	3 días	lun 10/05/21	mié 12/05/21		Tecnico
Revisar referencias	2 días	lun 31/05/21	mar 01/06/21	92FF	Tecnico

Figura 7 - Estimación de tiempo revisión final y documentación

5.3- Análisis de riesgos

Una vez que tenemos la planificación temporal, procedemos a analizar los riesgos al proyecto. Para categorizar los riesgos de la mejor manera posible, para cada riesgo se realizará una categorización en dos dimensiones: por un lado la probabilidad de ocurrencia, y por otro el impacto en caso de ocurrencia, ambos valores en una escala cualitativa con los valores “bajo”, “medio”, “alto” y “muy alto”. También se realiza un plan de contingencia y otro de mitigación para cada riesgo con objetivo de minimizar su impacto en caso de que llegue a ocurrir.

Riesgo	Probabilidad	Impacto	Contingencia	Mitigación
Pérdida de disponibilidad de algún integrante del proyecto (alumno o tutores)	medio	alto	Planificar una reunión con los tutores lo antes posible y modificar la planificación	Planificar reuniones semanales
Problema en la máquina de desarrollo que provoque la pérdida del progreso	Bajo	Muy alto	Recuperar las copias almacenadas	Almacenar los documentos y el código en varios dispositivos, además de en algún repositorio Git en la nube.
Errores en la planificación que aumenten la duración del proyecto	Medio	Alto	Reducir el alcance del proyecto	Intentar comenzar las tareas antes de su fecha planificada para mantener un colchón de seguridad
Cambios tardíos a los requisitos del proyecto	Medio	Alto	Modificar la planificación del proyecto para adaptarse a los cambios	Confirmar y congelar los requisitos durante el desarrollo de cada caso de uso

Figura 8 - Tabla de riesgos al proyecto

5.4- Presupuesto

Una vez que ya tenemos la estimación de la duración del proyecto y un análisis de riesgos, procedemos a calcular un presupuesto para el proyecto. Para ello creamos tres recursos que ejecutarán el proyecto (se omiten los tutores, por lo que su coste durante las reuniones de progreso no se tiene en cuenta):

- Técnico: Realiza todas las tareas de configuración previa, implementación y pruebas de cada caso de uso, revisión final del código y documentación del proyecto, además de estar presente en todas las reuniones de progreso. Tiene un coste (salario + costes administrativos) de 19€ la hora.
- Analista: Realiza las tareas de análisis y diseño de todos los casos de uso, además de estar presente en todas las reuniones de progreso. Tiene un coste de 50€ la hora.

- Gestor de Proyecto: Realiza la planificación inicial, así como las revisiones de la planificación al comienzo de cada caso de uso. Está presente en todas las reuniones de progreso y tiene un coste de 50€ la hora.

Con estos recursos, obtenemos las siguientes horas de trabajo para cada recurso, y su coste:

- Técnico: 493,5 horas, obtenemos un coste de 9.376€
- Analista: 89,4 horas, obtenemos un coste de 4.470€
- Gestor de Proyecto: 21,5 horas, obtenemos un coste de 1.075€

Esto nos da un coste total de 14.921€. Estimando que la empresa encargada de realizar este proyecto quiera obtener un margen de beneficios del 30% sobre el coste inicial del proyecto, añadiendo un 10% al valor de los costes debido a los riesgos, y añadiendo impuestos (únicamente IVA del 21%), realizamos los cálculos y obtenemos un presupuesto de 25.276€

Concepto	Cantidad
Costes iniciales	14.921 €
+ 10% riesgos	1.492 €
+ 30% beneficios	4.476 €
Subtotal	20.889 €
+ 21% IVA	4.387 €
TOTAL	25.276 €

Figura 9 - Presupuesto del proyecto

6- Modelo de análisis

A continuación se exponen los diversos modelos creados para el proyecto. Para este proyecto se ha realizado un diagrama de casos de uso (con el detalle para cada CU), un diagrama entidad-relación de un fragmento de la base de datos de Service Manager, un diagrama de Dominio, y diagramas de interacción para los casos de uso “Crear Incidencia”, “Modificar Incidencia” y “Consultar KPI”. Además, se ha realizado un diagrama de los servicios REST a exponer desde Service Manager para que el bot pueda acceder a ellos. Todos los diagramas han sido realizados utilizando la herramienta Astah Profesional, proporcionada por la Escuela.

6.1- Diagrama de Casos de Uso

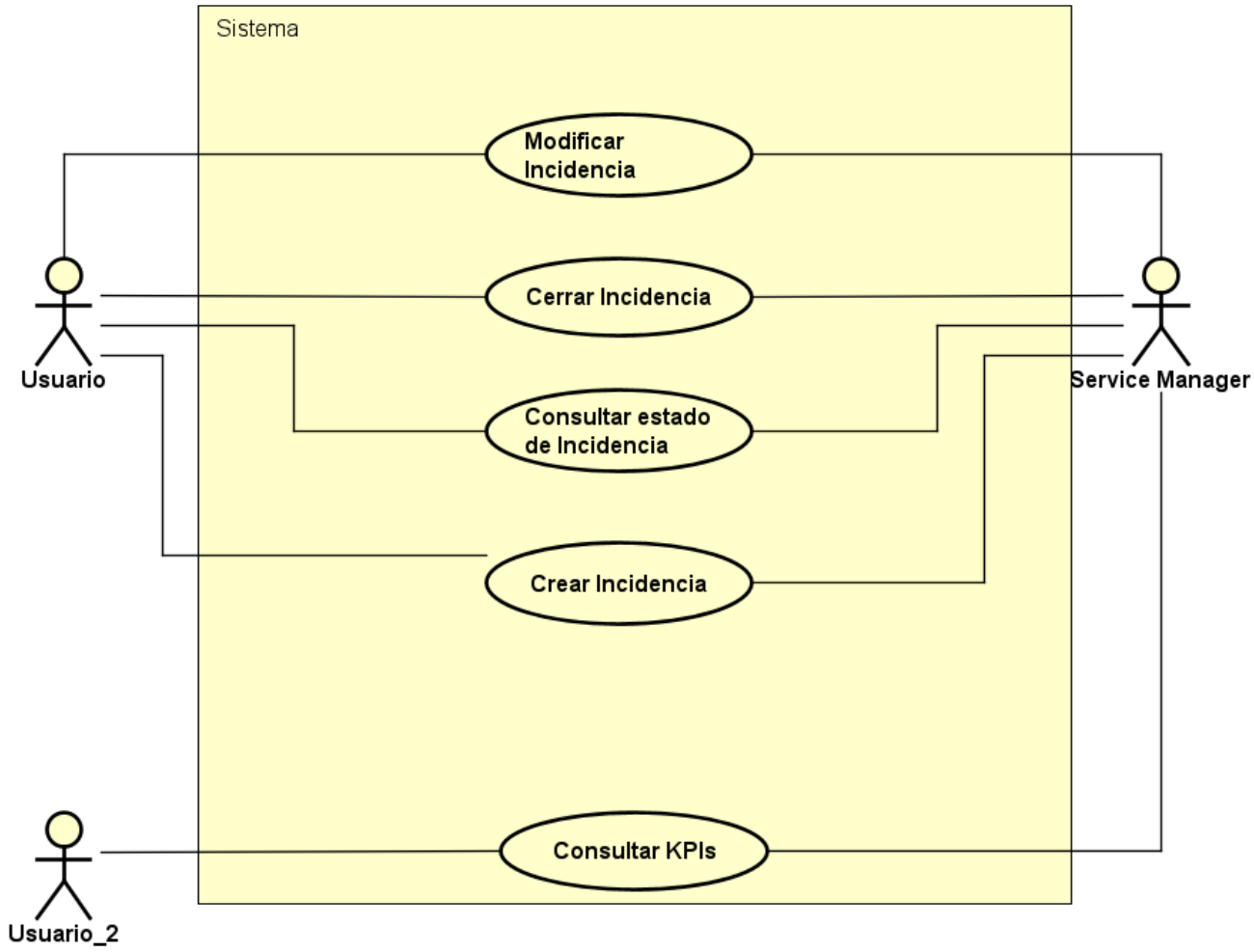

Figura 10- Diagrama de casos de uso

Como se puede observar, en el diagrama se presentan 2 actores. Por un lado tenemos un actor “Usuario”, que será el que realizará la mayoría de casos de uso. En segundo lugar, tenemos un actor “Usuario 2”, que realizará los casos de uso más técnicos, tales como la consulta de KPIs. Aunque se diferencian los usuarios, no existe una restricción de ningún tipo para ejecutar dichos casos de uso, únicamente se separan con objetivo de señalizar que en la práctica, la aplicación sería utilizada por dos tipos de usuarios o roles diferentes. También podemos apreciar que el sistema se comunica con otro sistema, Service Manager. Estas comunicaciones se harán exclusivamente a través del API REST de Service Manager.

6.2- Secuencia de los Casos de Uso

A continuación se detalla el proceso a seguir durante la ejecución de los diversos casos de uso, incluyendo escenarios adicionales:

6.2.1- Caso de Uso “Crear Incidencia”

ITEM	VALUE
UseCase	Crear Incidencia
Summary	Este caso de uso se ejecutará cuando el actor Usuario decida iniciar la creació n de una incidencia mediante el chat de Discord
Actor	Usuario, Service Manager, Usuario_2
Precondition	El usuario ha iniciado sesión en Discord y se encuentra en el servidor correcto
Postcondition	A no ser que el usuario cancele el proceso, se habrá creado una incidencia as ignada a dicho usuario
Base Sequence	1- Usuario inicia el caso de uso mediante un mensaje en el chat. 2- Sistema solicita a Usuario su nombre de usuario en Service Manager. 3- Usuario introduce su nombre de usuario (login). 4- Sistema comprueba que el usuario existe en Service Manager y que tiene p ermisos para crear Incidencias. 5- Sistema solicita a Usuario un título para la Incidencia. 6- Usuario introduce el título deseado para la Incidencia. 7- Sistema valida el título introducido. 8- Sistema solicita a Usuario una descripción de la Incidencia. 9- Usuario introduce una descripción de la incidencia. 10- Sistema valida la descripción de la incidencia. 11- Sistema muestra una lista de Servicios y solicita al usuario seleccionar el afectado a partir de su identificador. 12- Usuario introduce el identificador del sistema afectado. 13- Sistema verifica el identificador proporcionado. 14- Sistema solicita un valor de impacto, mostrando las opciones posibles. 15- Usuario selecciona un valor de impacto. 16- Sistema valida el valor introducido. 17- Sistema solicita un valor de urgencia, mostrando las opciones posibles. 18- Usuario selecciona un valor de urgencia. 19- Sistema valida el valor introducido. 20- Sistema crea la incidencia e informa al usuario del identificador de la incid encia creada correctamente. 21- FIN DEL CASO DE USO.
Branch Sequence	3b,6b,9b,12b,15b,18b - El usuario decide cancelar la operación. El caso de us o queda sin efecto. 4b- El usuario no existe, o bien no tiene permisos para crear incidencias. El si stema informa de ello y vuelve al paso 2. 7b- El título contiene caracteres no válidos. El sistema informa de ello y vuelve al paso 5. 10b- La descripción contiene caracteres no válidos. El sistema informa de ello y vuelve al paso 8. 13b- El identificador no se corresponde al de ningún servicio. El sistema infor ma de ello y vuelve al paso 11. 16b- El valor de impacto está fuera de los valores posibles. El sistema informa de ello y vuelve al paso 14. 19b- El valor de urgencia está fuera de los valores posibles. El sistema inform a de ello y vuelve al paso 17.
Exception Sequence	
Sub UseCase	
Note	

Figura 11- Detalle del Caso de Uso "Crear Incidencia"

6.2.2- Caso de Uso “Modificar Incidencia”

ITEM	VALUE
UseCase	Modificar Incidencia
Summary	Este caso de uso se ejecutará cuando el actor Usuario decida iniciar la modificación de una incidencia mediante el chat de Discord
Actor	Usuario, Service Manager, Usuario_2
Precondition	El usuario ha iniciado sesión en Discord y se encuentra en el servidor correcto
Postcondition	A no ser que el usuario cancele el proceso, se habrá modoficado una incidencia.
Base Sequence	1- Usuario inicia el caso de uso mediante un mensaje en el chat. 2- Sistema muestra una lista de los posibles campos que puede cambiar, y solicita a Usuario una lista de los que desea cambiar. 3- Usuario introduce la lista de campos a modificar. 4- Sistema valida la lista. 5- Para cada campo de la lista, Sistema solicita a Usuario el nuevo valor. 6- Para cada campo de la lista, Usuario introduce el nuevo valor. 7- Para cada valor introducido por el usuario, Sistema lo valida. 8- Sistema modifica la incidencia e informa al usuario de ello. 9- FIN DEL CASO DE USO.
Branch Sequence	3b,6b- El usuario decide cancelar la operación. El caso de uso queda sin efecto. 4b- La lista no es válida. El sistema informa de ello y vuelve al paso 2 7b- El valor no es válido. Sistema informa de ello y vuelve a pedir el valor.
Exception Sequence	
Sub UseCase	
Note	

Figura 12- Detalle del Caso de Uso "Modificar Incidencia"

6.2.3- Caso de Uso “Consultar estado de Incidencia”

ITEM	VALUE
UseCase	Consultar estado de Incidencia
Summary	Este caso de uso se ejecutará cuando el actor Usuario decida iniciar la consulta del estado de una incidencia mediante el chat de Discord
Actor	Usuario, Service Manager, Usuario_2
Precondition	El usuario ha iniciado sesión en Discord y se encuentra en el servidor correcto
Postcondition	A no ser que el usuario cancele el proceso, el usuario habrá obtenido el estado de la incidencia indicada.
Base Sequence	1- Usuario inicia el caso de uso mediante un mensaje en el chat. 2- Sistema solicita a Usuario el identificador de la incidencia en Service Manager 3- Usuario introduce el identificador de la incidencia 4- Sistema obtiene la Incidencia de Service Manager 5- Sistema informa del estado de la incidencia. 6- FIN DEL CASO DE USO.
Branch Sequence	3b- El usuario decide cancelar la operación. El caso de uso queda sin efecto. 4b- La incidencia no existe. El sistema informa de ello y vuelve al paso 2
Exception Sequence	
Sub UseCase	
Note	

Figura 13- Detalle del Caso de Uso "Consultar estado de Incidencia"

6.2.4- Caso de Uso “Cerrar Incidencia”

ITEM	VALUE
UseCase	Cerrar Incidencia
Summary	Este caso de uso se ejecutará cuando el actor Usuario decida iniciar la consulta del estado de una incidencia mediante el chat de Discord
Actor	Usuario, Service Manager, Usuario_2
Precondition	El usuario ha iniciado sesión en Discord y se encuentra en el servidor correcto
Postcondition	A no ser que el usuario cancele el proceso, la incidencia se habrá cerrado.
Base Sequence	1- Usuario inicia el caso de uso mediante un mensaje en el chat. 2- Sistema solicita a Usuario el identificador de la incidencia en Service Manager 3- Usuario introduce el identificador de la incidencia 4- Sistema obtiene la Incidencia de Service Manager 5- Sistema informa de los datos de la incidencia y pide confirmación para cerrarla. 6- Usuario confirma el cierre de la incidencia. 7- Sistema cierra la incidencia e informa de ello al usuario. 8- FIN DEL CASO DE USO.
Branch Sequence	3b,6b- El usuario decide cancelar la operación. El caso de uso queda sin efecto. 4b- La incidencia no existe. El sistema informa de ello y vuelve al paso 2
Exception Sequence	
Sub UseCase	
Note	

Figura 14- Detalle del Caso de Uso "Cerrar Incidencia"

6.2.5- Caso de Uso “Consultar KPIs”

ITEM	VALUE
UseCase	Consultar KPIs
Summary	Este caso de uso se ejecutará cuando el actor Usuario_2 decida iniciar lacons ulta de uno o varios KPIs (Key Performance Indicator) mediante el chat de Dis cord
Actor	Usuario, Service Manager, Usuario_2
Precondition	El usuario ha iniciado sesión en Discord y se encuentra en el servidor correcto
Postcondition	A no ser que el usuario cancele el proceso, el usuario habrá obtenido los valo res de los KPIs solicitados
Base Sequence	1- Usuario inicia el caso de uso mediante un mensaje en el chat. 2- Sistema muestra a Usuario la lista de KPIs disponibles y solicita escoger los deseados. 3- Usuario introduce en forma de lista separada por comas los KPIs deseados. 4- Sistema valida la lista de KPIs introducida. 5- Sistema obtiene los KPIs de Service Manager y los muestra uno a uno. 6- FIN DEL CASO DE USO.
Branch Sequence	3b- El usuario decide cancelar la operación, el caso de uso queda sin efecto 4b- La lista contiene algún elemento no válido. El sistema informa del element o o elementos no válidos y vuelve al paso 2.
Exception Sequence	
Sub UseCase	
Note	

Figura 15- Detalle del caso de uso "Consultar KPIs"

6.3- Diagrama simplificado de la base de datos de Service Manager

En el siguiente diagrama se presenta un fragmento muy simplificado de la base de datos de Service Manager.

Como la base de datos original tiene cientos de tablas con decenas de campos para cada tabla, se ha simplificado el diagrama para mostrar solo las tablas que utilizaremos a lo largo del proyecto, indicando dentro de estas tablas los campos más importantes. De esta manera tendremos una mejor percepción de las entidades con las que deberemos trabajar posteriormente.

Como se indica en el diagrama, el nombre de las tablas no es muy intuitivo y es necesario aclarar el propósito de algunas de ellas: La tabla “Probsummary” es la tabla de Incidencias en Service Manager, mientras que la tabla “Incident” es la tabla de Interacciones. Por otro lado, la tabla “Activity” almacena cambios en las Incidencias, tales como cambios de datos, de estado, reasignaciones…

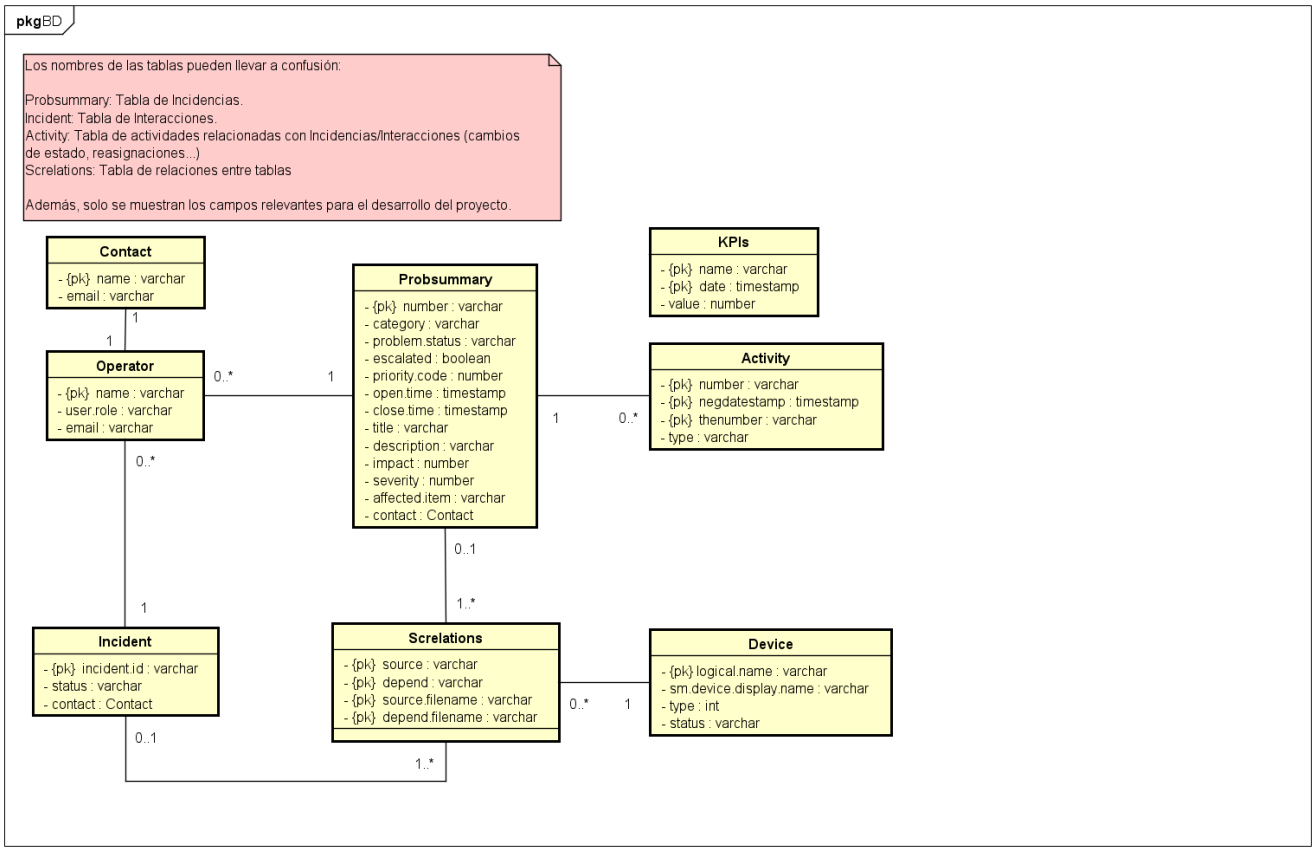

Figura 16- Simplificación de la base de datos de SM

6.4- Modelo de dominio de la aplicación

A partir de los diagramas de casos de uso y entidad-relación, creamos el modelo de dominio que utilizaremos en el proyecto.

Se ha optado por incluir solo las clases “Incident” (en representación de las Incidencias) y KPI (en representación de los KPIs) debido a que en el caso de otros posibles elementos de dominio como podrían ser los CIs o los operadores se comprueba que no se realizarían modificaciones en dichos campos, por lo que estaríamos creando un elemento de dominio para utilizarlo únicamente como registro, lo cual es incorrecto.

En el caso del KPI, esta situación no se da debido a la forma en la que se almacena la fecha de obtención del KPI en Service Manager. Como queremos devolver la fecha en un formato más legible para el usuario (dd/mm/yyyy hh:mm) el método de devolución de la fecha la procesará antes de devolverla para darle el formato deseado.

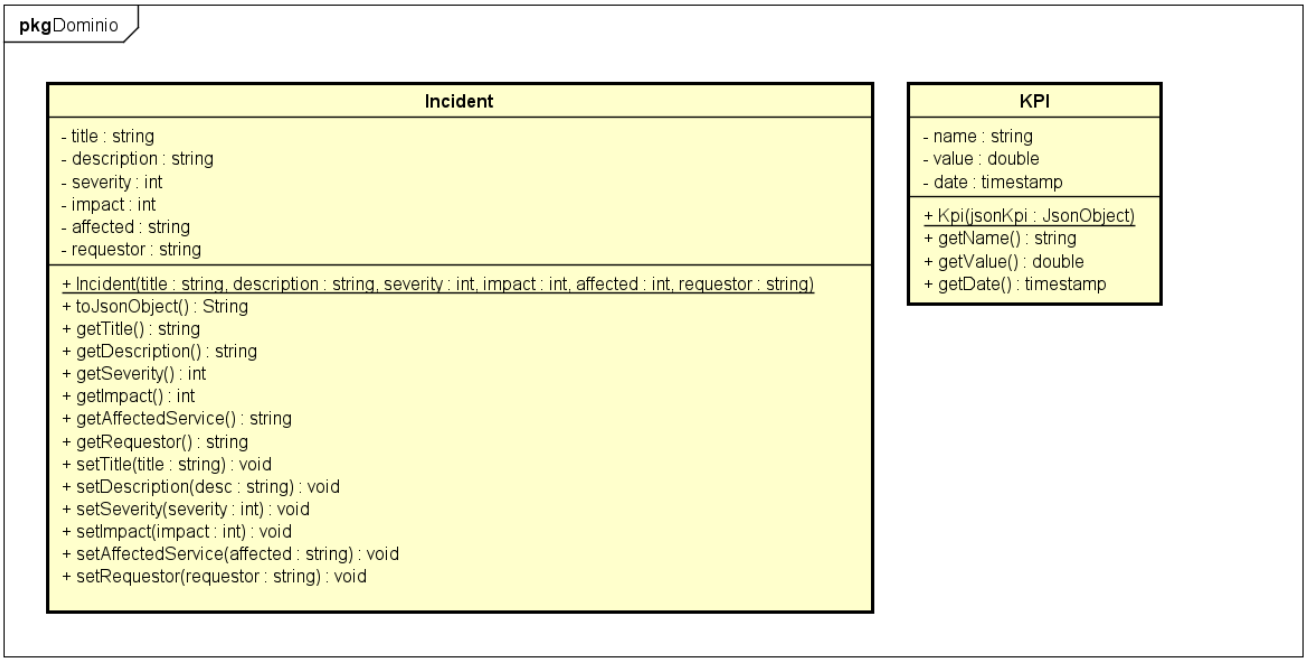

Figura 17- Modelo de dominio de la aplicación

6.5- Arquitectura empleada en la aplicación

Una vez que tenemos los diagramas anteriores, procedemos a realizar el diagrama de la arquitectura que tendrá la aplicación. Como se puede observar, para el desarrollo de la aplicación se utilizará el patrón MVC. Por un lado tenemos una vista (view) que se encarga de recibir y enviar mensajes a través de Discord. Esta clase se comunica con el controlador (controller) que contendrá la lógica de los casos de uso, utilizando los modelos definidos en la capa de modelo (model).

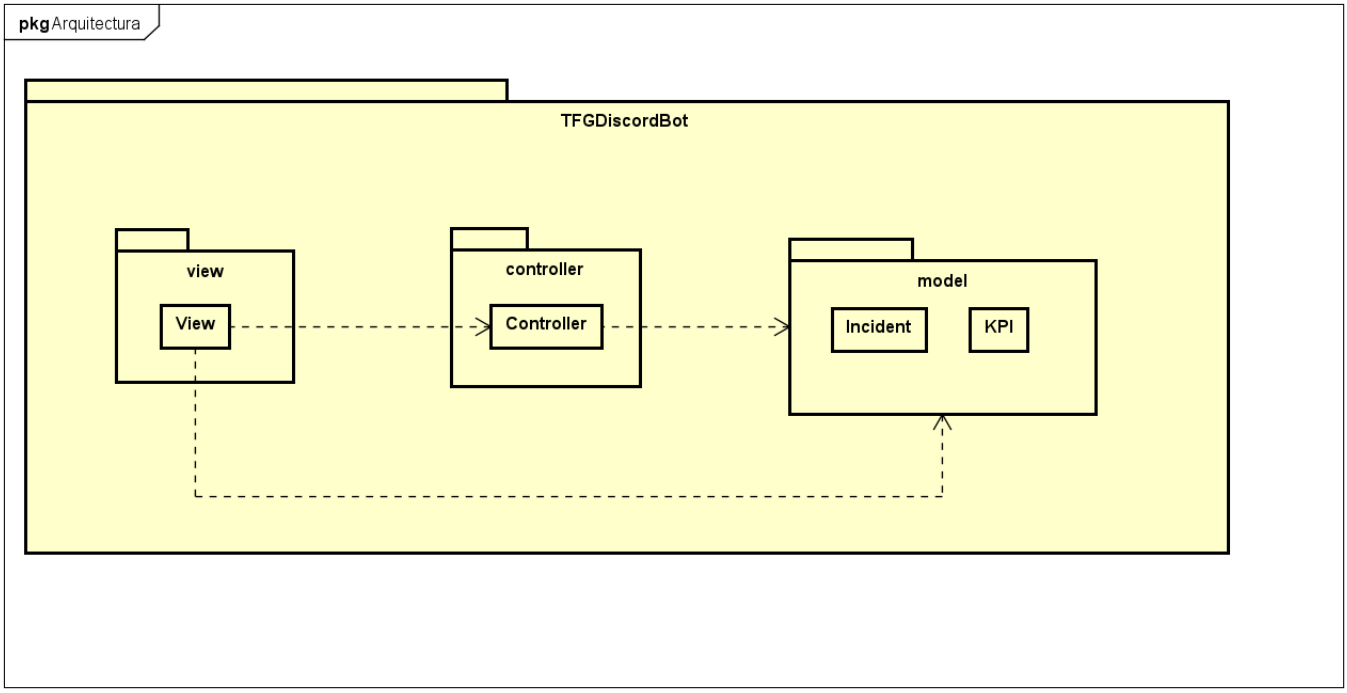

Figura 18 - Arquitectura genérica propuesta

6.6- Diagrama de recursos REST expuestos en Service Manager

Para finalizar con los diagramas, creamos el diagrama de recursos REST que expondremos desde Service Manager, a los que accederá el bot de Discord. Cabe destacar que, al igual que en el diagrama entidad-relación, la lista de atributos de los recursos es mayor a la que se muestra en el diagrama, ya que solo se muestran aquellos que nos resultarán relevantes para el proyecto.

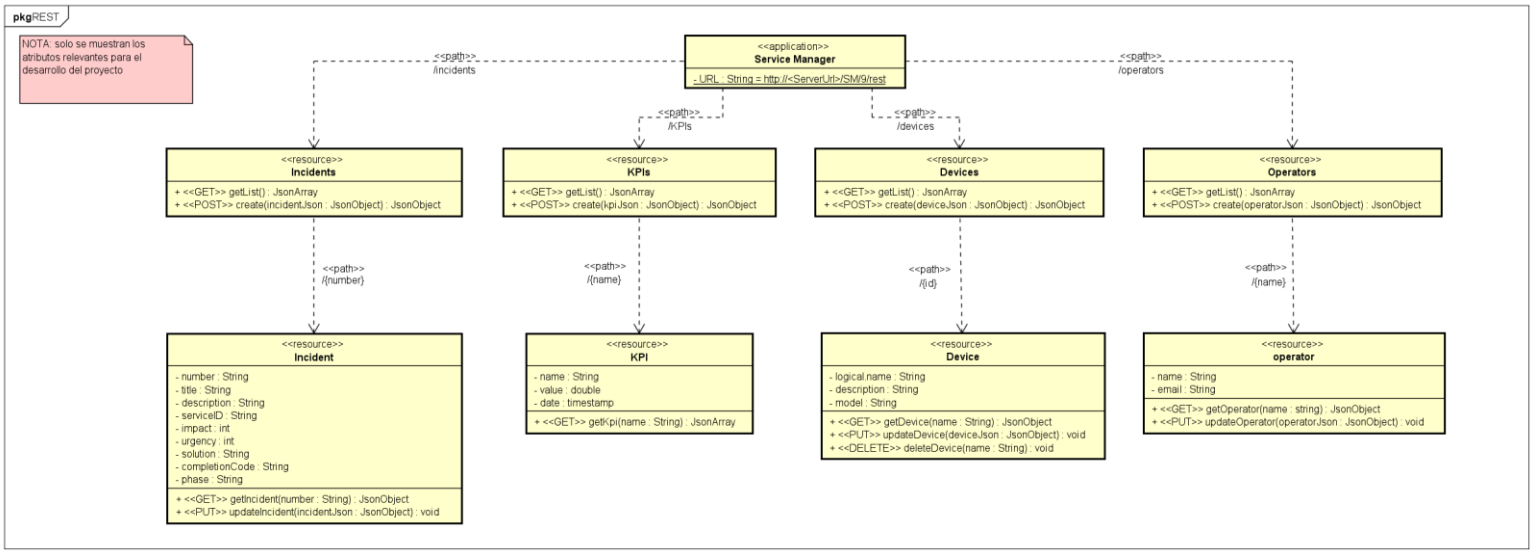

Figura 19 - Diagrama REST de la aplicación

7- Diseño

Ahora que ya hemos realizado un análisis exhaustivo del proyecto a realizar, definiremos los aspectos de diseño: Realizaremos los diagramas de secuencia para cada caso de uso, y escogeremos una metodología adecuada para el proyecto.

7.1- Metodología

Para el desarrollo del trabajo, se ha comenzado en primer lugar con periodo de entrenamiento de uso de la herramienta Service Manager, con objetivo de entender el funcionamiento de la aplicación, los procesos ITIL involucrados en la aplicación, instalar la herramienta en una máquina virtual CentOS 8, desplegar el cliente web de la aplicación en un servidor Tomcat, e instalar el cliente en una máquina Windows. Para más detalles acerca de esta primera fase, consultar el apartado 8.1: “Configuración inicial”.

Tras la primera fase, se utiliza una metodología de desarrollo iterativo, en la que se va realizando el análisis, diseño, implementación y pruebas de cada uno de los casos de uso, con reuniones semanales con el tutor externo para verificar y validar el progreso realizado durante la semana, lo que permite ir refinando los diversos artefactos producidos a lo largo de las semanas, eliminando errores, simplificando flujos excesivamente complejos, o añadiendo algunos requisitos extra para simplificar el uso del bot.

Las iteraciones realizadas han sido similares para cada caso de uso, y se han realizado de la siguiente forma:

- Primera iteración: Análisis y diseño del caso de uso, implementación básica sin control de errores (únicamente flujo exitoso en situación ideal), con una duración aproximada de unos 7 días.
- Segunda iteración: Refinado de los diagramas realizados en la primera iteración, implementación de todos los flujos posibles en el caso de uso y control de errores adecuado, comienzan las pruebas básicas. Duración aproximada de 10 días.
- Tercera iteración: Revisión final de los diagramas realizados, refactorización del código y pruebas exhaustivas para detectar y resolver el mayor número posible de fallos. Revisión de requisitos para comprobar que se han cumplido correctamente. Duración aproximada de 8 días.

Antes de comenzar el desarrollo del proyecto, se realizó una fase inicial de configuración y aprendizaje de las herramientas que se utilizaron a lo largo del proyecto, con objetivo de evitar retrasos a lo largo de la realización del proyecto. Los detalles de esta fase previa se expondrán posteriormente.

Para el desarrollo del bot se ha usado exclusivamente el lenguaje de programación *Python 3* en un entorno Linux (CentOs8), aunque se ha utilizado también JavaScript en Service Manager para la realización de varios scripts de los que se hablará en la sección de implementación del caso de uso “Consultar KPIs”.

Como Base de Datos para Service Manager, se ha utilizado PostgreSQL versión 12.

Para los diversos diagramas realizados en las fases de análisis y diseño, se ha utilizado la herramienta *Astah Profesional*, proporcionada por la Escuela.

Para el análisis de tiempo y coste se ha utilizado la herramienta *Microsoft Project*, proporcionada por la Escuela como parte del paquete Office365.

Para la creación de este y otros documentos involucrados en el proyecto, se ha utilizado la herramienta *Microsoft Word*, proporcionada por la Escuela como parte del paquete Office365, debido a su facilidad de uso para añadir imágenes o dar formato a las páginas, al contrario que otras aplicaciones como LaTeX.

Para el almacenamiento y control de versiones del código fuente del proyecto, se ha utilizado la plataforma GitHub, con el repositorio “TFG_DiscordBot” [1]

7.2- Diagramas de interacción

Antes de pasar a la fase de implementación del proyecto, realizamos varios diagramas de interacción para los casos de uso “Crear Incidencia”, “Modificar Incidencia” y “Consultar KPI”, con objetivo de conocer el flujo de operaciones a utilizar a lo largo de los casos de uso.

Los diagramas se han realizado a partir de los detalles de los casos de uso realizados en el apartado 6.2. Se ha reflejado el flujo de eventos, con los escenarios alternativos dependiendo de la entrada del usuario y del estado de la base de datos.

7.2.1- Caso de Uso “Crear Incidencia”

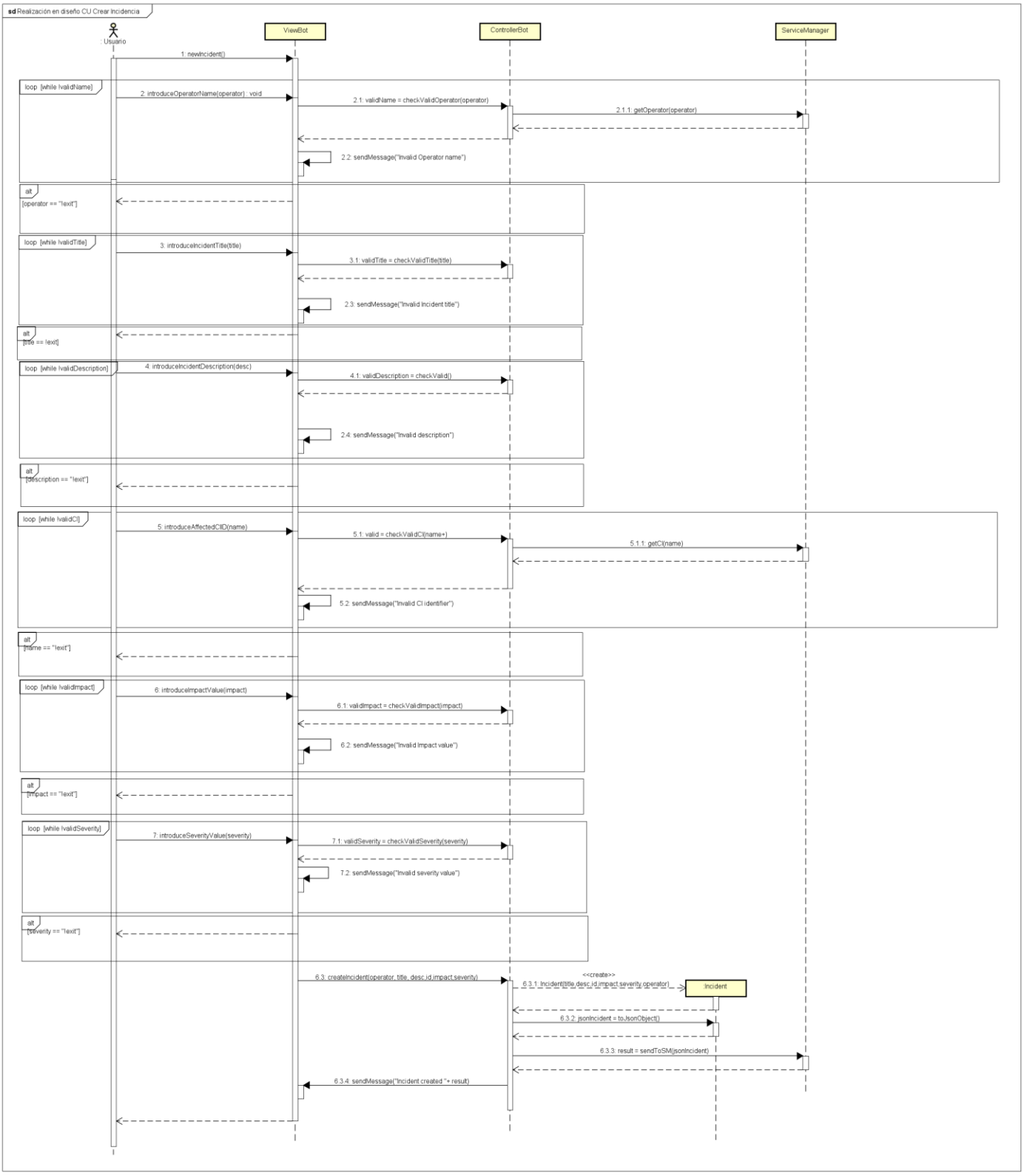

Figura 20 - Diagrama de Interacción CU Crear Incidencia

7.2.2- Caso de Uso “Modificar Incidencia”

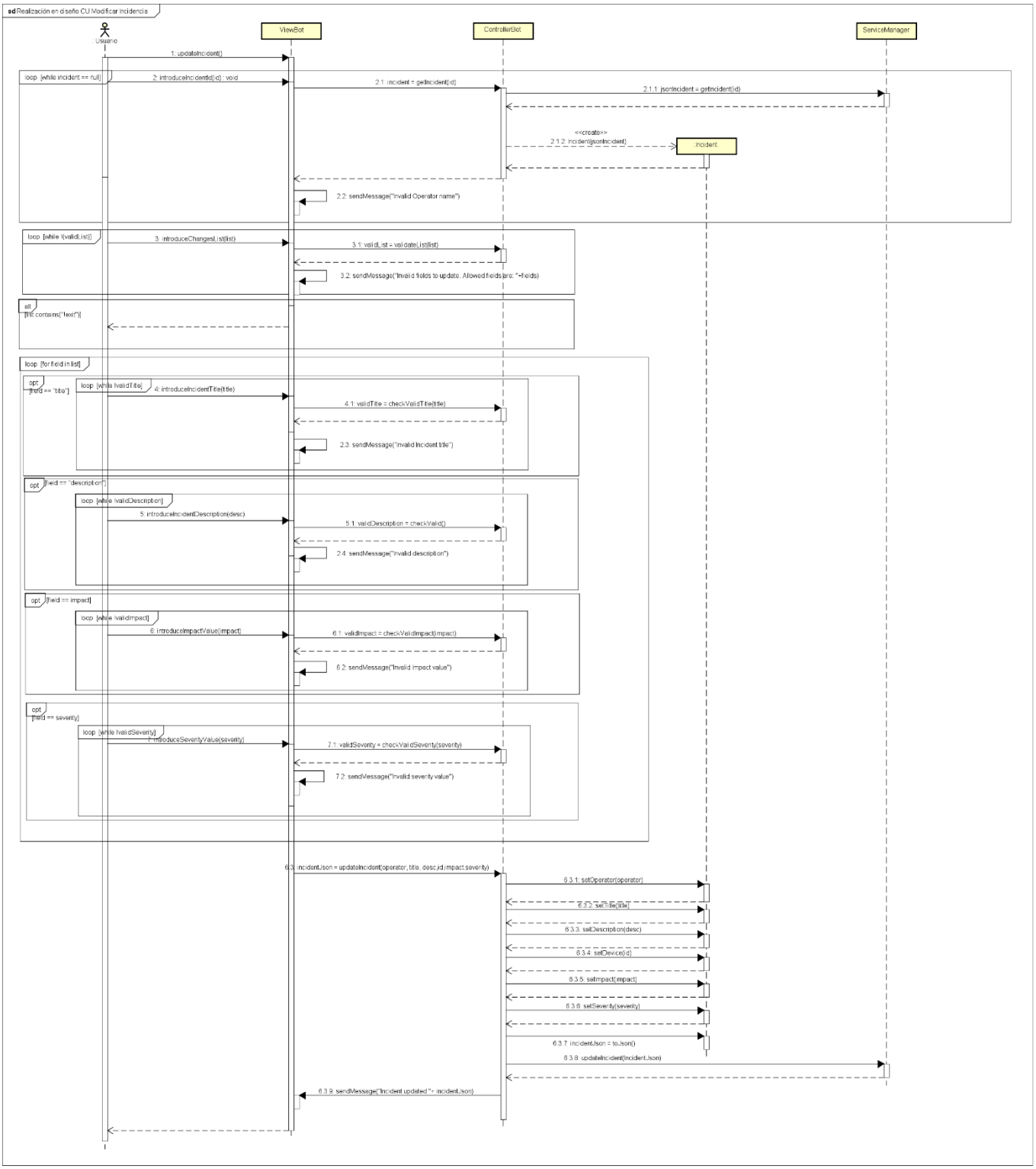

Figura 21 - Diagrama de Interacción CU Modificar Incidencia

7.2.3- Caso de Uso “Consultar KPI”

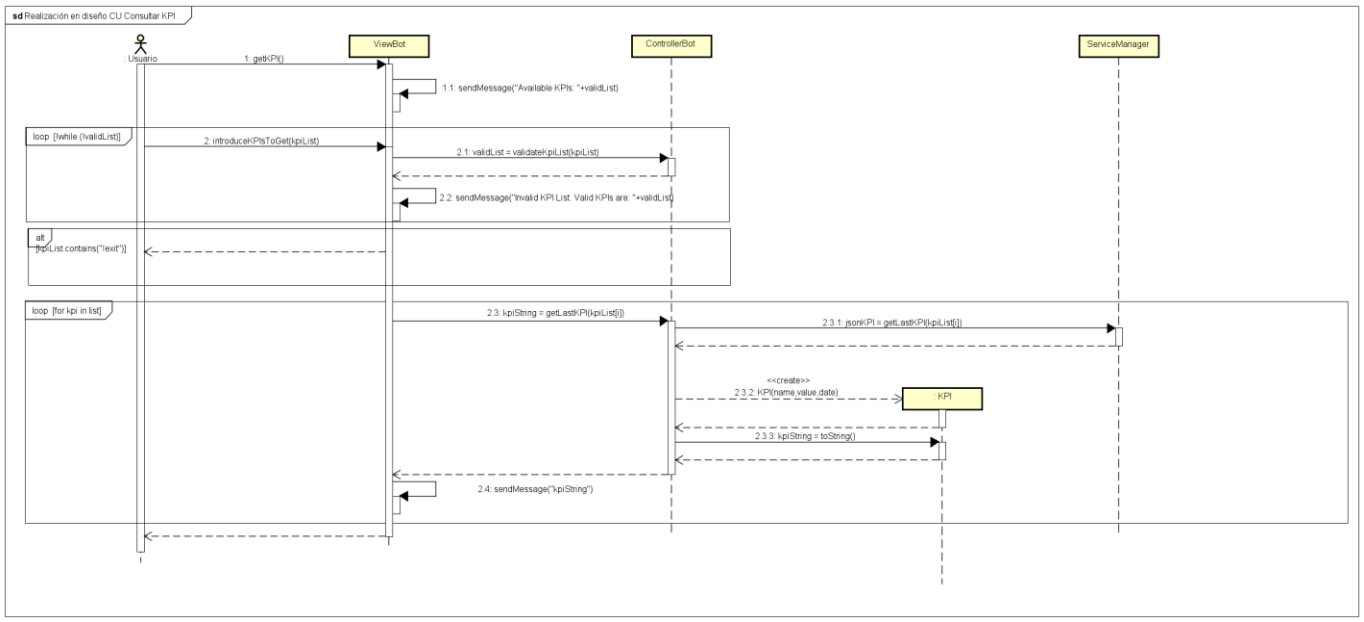

Figura 22 - Diagrama de Interacción CU Consultar KPI

8- Implementación

8.1- Fase Previa al comienzo del desarrollo del Bot.

Antes de comenzar el desarrollo propio del proyecto, se dedican varias semanas a la "Configuración inicial". Esto es, la instalación y configuración del entorno virtual en el que se instalará el servidor de Service Manager, la instalación del propio Service Manager, el despliegue de su cliente web y la instalación del cliente Windows, y los aprendizajes tanto de la aplicación Service Manager en sus dos clientes, como del desarrollo de bots de Discord utilizando Python. Esta fase se ha incluido en la planificación inicial del proyecto, en la sección de "Configuración previa"

8.1.1- Instalación de la Máquina Virtual

El primer paso realizado fue el de instalar una máquina virtual CentOs 8, que se utilizará como servidor en el que se ejecutarán los servidores de Service Manager, Tomcat y postgreSQL. Para obtener la imagen del sistema operativo, accedemos a la página de descargas de CentOS [2] y descargar la imagen ISO de CentOs 8 64 bits, versión DVD.

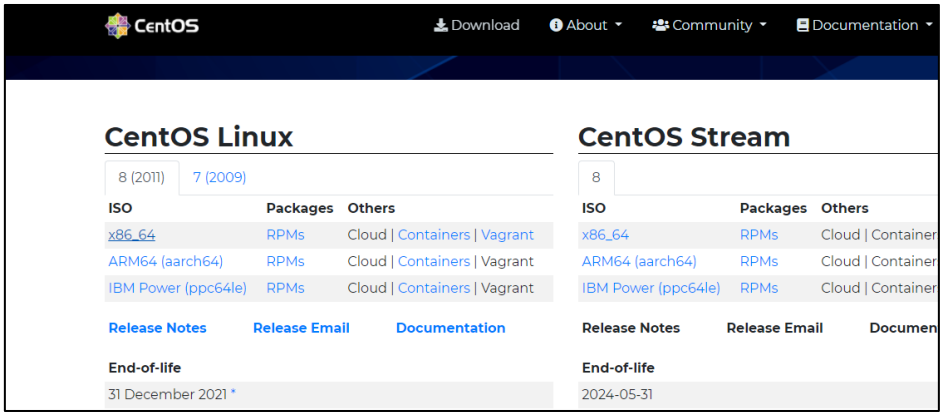

Figura 23- Página de descargas de CentOs

A continuación, creamos una máquina virtual usando Oracle VM VirtualBox. Las características de la máquina virtual son las siguientes:

- Nombre: CentOS TFG
- Tipo: Linux (other)
- Tamaño de memoria: 8192 MB
- Tipo de disco duro: VDI
- Almacenamiento en unidad física: Reservado dinámicamente.

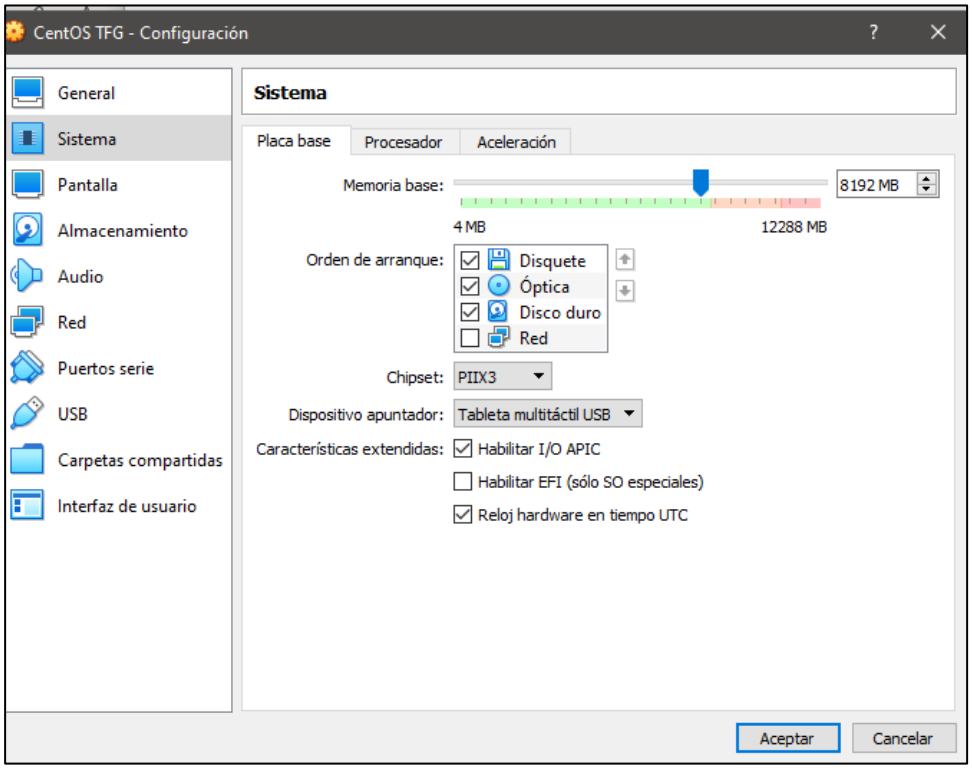

Figura 24- Configuración de la máquina virtual

Tras esto, Se añade una unidad de CD a la máquina virtual con la imagen descargada previamente, y se comprueba que en el orden de arranque esté antes la unidad de CD que el disco duro, iniciamos la máquina virtual, y ejecutamos el instalador.

Figura 25- Primera pantalla tras el arranque de la máquina virtual

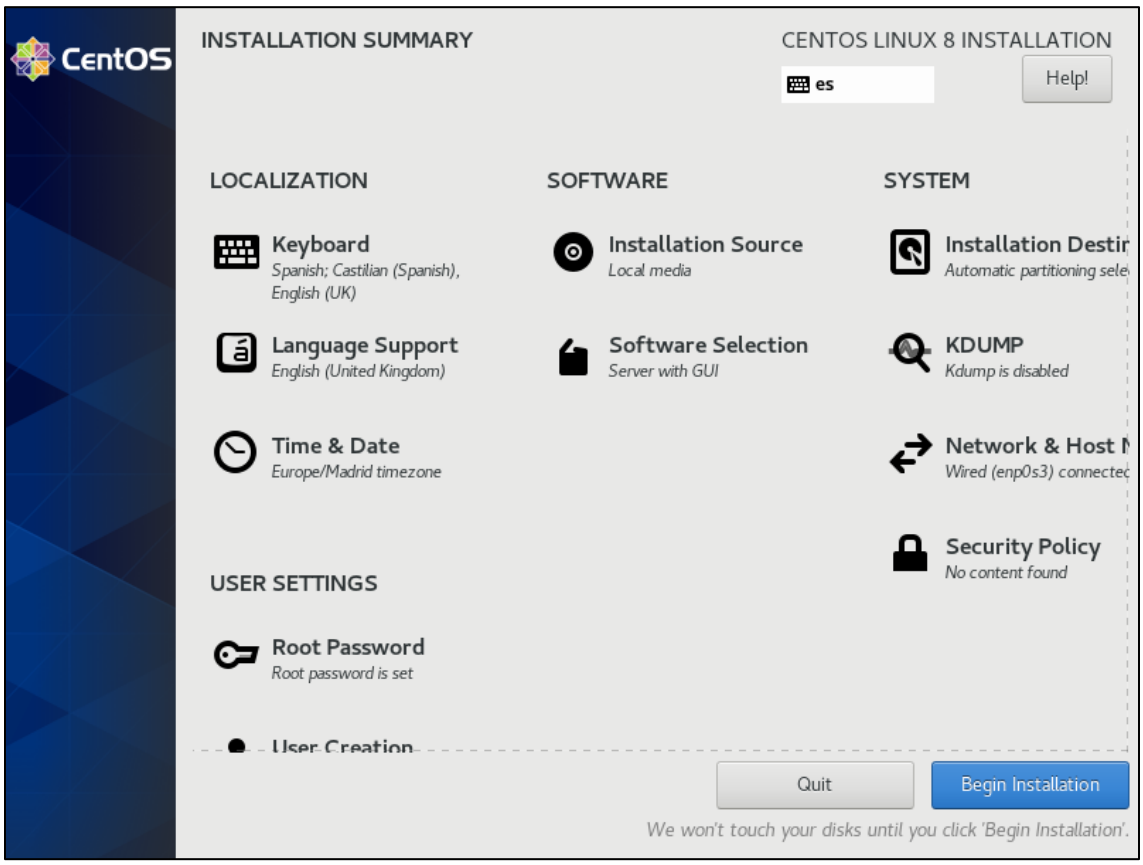

Figura 26- Configuración del instalador de CentOs8

Durante la instalación, seleccionamos el teclado en Español, establecemos la zona horaria, y creamos la contraseña para root (iR+uVqcgcf) y el usuario administrador (jcarlos, con contraseña TFG20-21). Como queremos comprobar rápidamente el acceso al cliente web cuando esté desplegado, instalamos la opción de “Server with GUI”, que nos permitirá utilizar un navegador gráfico.

Tras haber seleccionado la configuración deseada, iniciamos la instalación, con el resto de las características por defecto, y esperamos a que finalice. Tras finalizar la instalación reiniciamos el sistema, aceptamos el acuerdo de licencia, e iniciamos el sistema de manera satisfactoria.

8.1.2- Instalación de PostgreSQL 12 [3]

Ahora que ya tenemos el sistema operativo correctamente instalado, abrimos un emulador de terminal para instalar PostgreSQL12. En primer lugar ejecutamos el comando ***dnf module list postgresql*** (dnf es el instalador de paquetes de CentOs8) para listar todas las versiones disponibles, obteniendo que podemos instalar 3 versiones: 9.6, 10 y 12, siendo la que se instala por defecto la 10. Como queremos la versión 12, la habilitamos mediante el comando ***dnf module enable postgresql:12*** y a continuación lo instalamos con con ***dnf install postgresql-server ()***

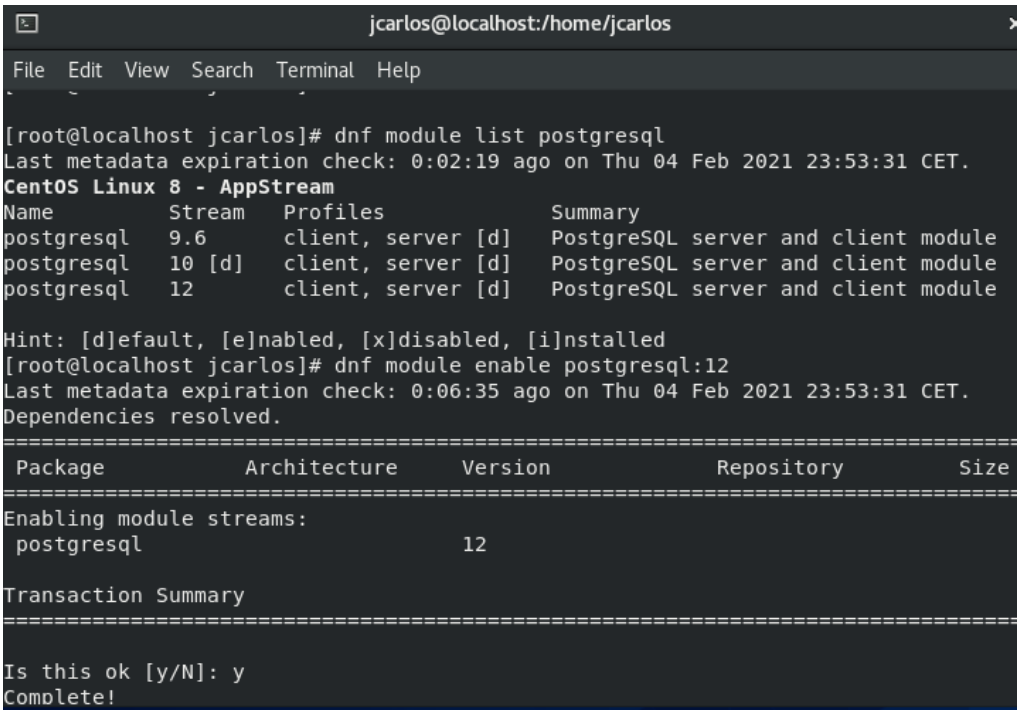

Figura 27- Selección de la versión de postgreSQL a instalar

Tras la instalación, ejecutamos el comando **postgresql-setup –initdb**, e iniciamos el servicio con el comando **systemctl start postgresql**, cambiamos a la cuenta de postgres con **sudo -i -u postgres**, y creamos una base de datos de prueba (**createdb prueba**), a la que accedemos (**psql -d prueba**) Allí creamos una tabla sencilla, añadimos varios registros, y los obtenemos mediante varias consultas, comprobando así que funciona correctamente.

8.1.3- Instalación de Service Manager [5]

Comenzamos por instalar Perl en nuestra máquina virtual, debido a que es utilizado por el instalador (**sudo yum install perl**). A continuación, creamos el directorio en el que instalaremos el servidor de Service Manager. En nuestro caso, lo crearemos en /home/jcarlos, y lo llamaremos SM9.7 (**mkdir /home/jcarlos/SM9.7**).

A continuación, ejecutamos el fichero proporcionado por el tutor externo, que contiene el instalador de Service Manager (./setupLinuxX64-9.70.bin), aceptamos los diversos contratos, e indicamos el directorio de instalación.

Una vez que la instalación ha finalizado de forma exitosa, accedemos de nuevo al usuario postgres, y creamos una base de datos llamada “sm”, que será la utilizada por Service Manager (**createdb sm**)

Ahora tenemos que permitir el acceso a Service Manager a la base de datos, para ello modificamos el fichero /var/lib/psql/data/pg_hba.conf para añadir el acceso a la nueva base de datos

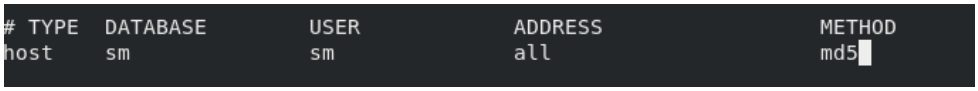

Figura 28- Línea añadida en el fichero pg_hba.conf en el que se permite el acceso al usuario sm

Hemos permitido el acceso a la base de datos “sm” por parte del usuario “sm”, que todavía no existe. Lo que haremos a continuación será crear este usuario de postgres y darle privilegios sobre la base de datos. Esto se hace con las líneas “**create user sm;**” y “**grant ALL on database “sm” to sm;**”. Para finalizar, cambiamos el propietario de la base de datos al nuevo usuario mediante la línea “**ALTER DATABASE “sm” OWNER TO sm;**”, y reiniciamos el servicio (**systemctl restart postgresql**). Ahora ya podemos conectar nuestro servidor de Service Manager con la base de datos creada. Para ello, ejecutamos el script “configure.sh” que se ha creado durante la instalación de Service Manager.

El script nos va pidiendo los datos necesarios (puerto, tipo de entorno producción/pruebas, tipo de base de datos, y características de la base de datos, tales como el host, el puerto y el nombre). Tras un primer error debido a la falta de una librería que instalamos, ejecutamos el script de nuevo, terminando de manera correcta.

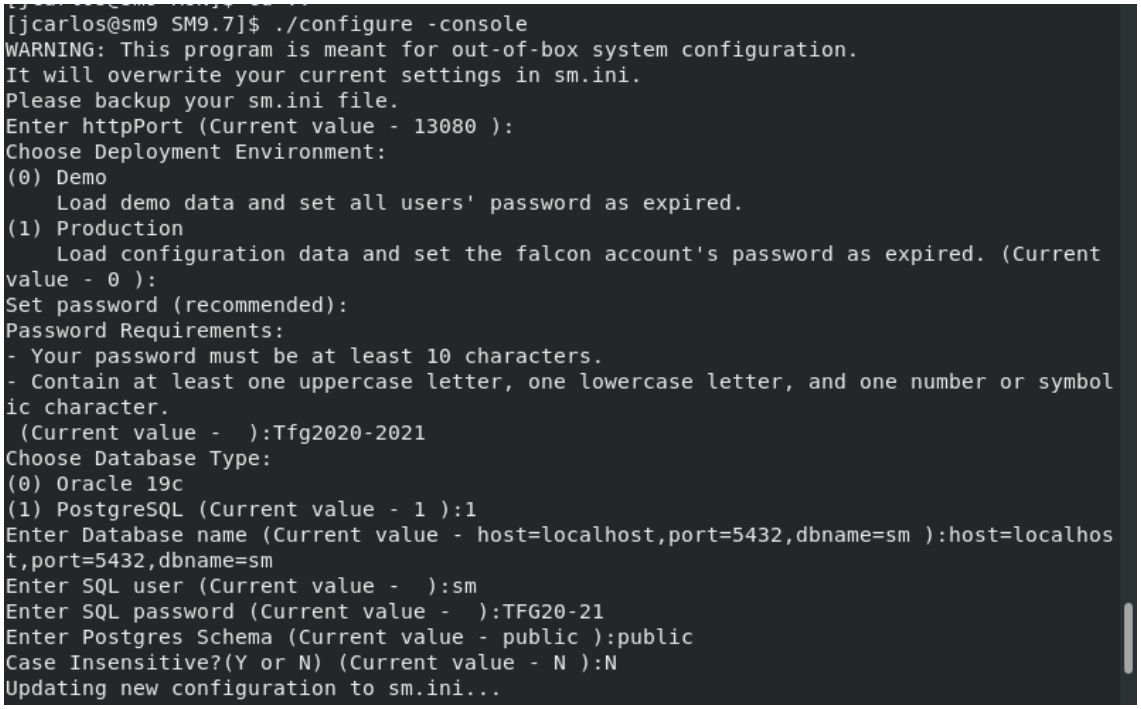

Figura 29- Ejecución del script de configuración de SM

Tras finalizar la configuración, aceptamos la creación de datos de prueba para la base de datos, lo cual crea múltiples tablas y las rellena con los campos de prueba necesarios. Esto tarda varios minutos debido a la inmensa cantidad de tablas y registros que se crean durante el proceso.

Una vez que ha finalizado la creación de tablas y registros, iniciamos el servidor mediante el script "**smstart**", instalado dentro del directorio "RUN" de Service Manager. Con esto ha quedado correctamente configurado el servidor de Service Manager, ya solo necesitamos un cliente con el que acceder.

Para que dicho cliente pueda conectarse al servidor, necesitamos configurar el firewall de la máquina virtual para permitir las conexiones en el puerto **13080**. Utilizando el comando "**firewall-cmd**", creamos el servicio, le asignamos el puerto y el protocolo, lo añadimos a la lista, y recargamos el cortafuegos para asegurarnos de que el nuevo servicio se ha añadido.

```
[root@sm9 ~]# firewall-cmd --permanent --new-service=sm
success
[root@sm9 ~]# firewall-cmd --permanent --service=sm --add-port=13080/tcp
success
[root@sm9 ~]# firewall-cmd --permanent --add-service=sm
success
[root@sm9 ~]# firewall-cmd --reload
success
[root@sm9 ~]# firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp0s3
  sources:
  services: cockpit dhcpv6-client sm ssh
  ports:
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
[root@sm9 ~]#
```

Figura 30- Configuración del cortafuegos

Ahora procedemos a instalar el cliente Windows de Service Manager. En este caso, el instalador consiste en un archivo .exe proporcionado por el tutor externo que realiza la instalación de forma automática. Tras la instalación, iniciamos el cliente, configuramos la conexión con el servidor, e iniciamos sesión con el operador de pruebas (falcon). Tras el primer inicio de sesión el sistema nos pide cambiar la contraseña del operador, por lo que la cambiamos a "**Tfg20202021**".

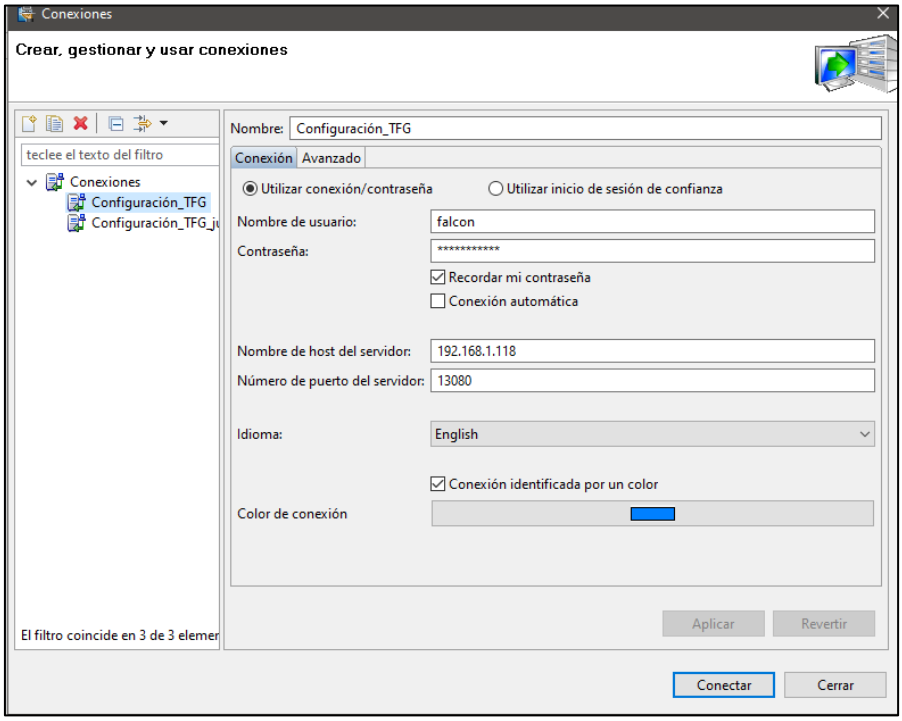

Figura 31- Configuración de la conexión en el cliente de Service Manager

Aunque ya podemos acceder a Service Manager a través del **cliente pesado** Windows, también desplegaremos el **cliente web** de Service Manager, ya que nos será útil para acceder a alguna funcionalidad que no está disponible en el cliente Windows, como por ejemplo la visualización y configuración de flujos de estados para cada registro en Service Manager.

Comenzamos por instalar Tomcat 9 en la máquina virtual (**sudo dnf install Tomcat**). A continuación, instalamos **Java Zulu 8.52 JDK**. Una vez que tenemos tanto el Java como el Tomcat, añadimos los puertos que utilizaremos (8080 y 8443 una vez que habilitemos la consexión segura ssl) utilizando firewalld. Movemos el fichero .war proporcionado por el tutor externo al directorio webapps de Tomcat. Tras esto recargamos Service Manager y Tomcat, consiguiendo acceder al cliente web de manera satisfactoria.

8.1.4- Aprendizaje de uso de varios comandos de Service Manager.

Antes de comenzar con el desarrollo del bot, es necesario **conocer la herramienta** que se va a utilizar, con objetivo de determinar la manera de realizar las acciones necesarias para el correcto funcionamiento del futuro bot, sin saltarnos ninguna etapa en los procesos.

En primer lugar, se realiza una exploración de varios **comandos** utilizados por Service Manager que nos serán útiles. Estos comandos se escriben en la parte superior izquierda de la aplicación, tal y como se muestra en la imagen:

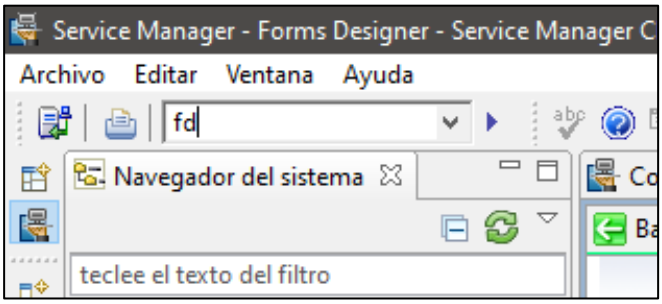

Figura 32- Campo de texto para comandos en Service Manager

Los comandos que se han estudiado han sido los siguientes:

a. **comp**: Permite modificar los datos de la compañía (nombre, dirección, ciudad…) y otros datos de configuración como la longitud de las contraseñas de los operadores, el formato de la fecha y hora, o las abreviaturas para los meses. Para explorar la vista, se modifican algunos parámetros
b. **contacts**: Permite tanto la búsqueda de un contacto a partir de su nombre completo como la creación de nuevos contactos. Para investigar esta vista más a fondo, creamos dos contactos cuyos operadores utilizaremos posteriormente a lo largo del proyecto: “jcgd” y “bot”.
c. **operator**: De manera similar a contacts, permite buscar o crear operadores. Todo operador debe tener antes un contacto. Creamos un operador para cada contacto

creado anteriormente, uno con rol de administradores, lo que les proporcionará acceso a toda la funcionalidad de la aplicación.

d. **df**: Permite crear formularios o editar formularios ya existentes. El nombre del formulario en uso se muestra en la parte inferior derecha de la aplicación. Mediante este comando podemos crear nuevos formularios con diversos campos tanto para la creación como para la visualización de los datos de varios registros. Lo utilizaremos a lo largo del desarrollo del proyecto para crear los formularios que nos permitan visualizar los datos de los KPIs creados posteriormente.
e. **db**: Este comando nos lleva a la vista en la que podemos añadir nuevas tablas o modificar campos de una tabla existente la base de datos. Lo utilizaremos en el proyecto para crear la tabla para los KPIs que se calcularán y almacenarán en la base de datos.
f. **sd**: Permite crear scripts en JavaScript que utilicen y/o generen entradas de la base de datos. Crearemos varios para el cálculo y registro de los diversos KPIs que utilizaremos en el proyecto.
g. **sch**: Nos lleva a la vista del planificador de Service Manager. Aquí podemos crear planificadores y añadirles tareas a realizar cada cierto tiempo (scripts a ejecutar). Utilizaremos un planificador para ejecutar de forma automática nuestros scripts calculadores de KPIs cada cierto tiempo, de manera que dichos indicadores siempre se mantengan actualizados.
h. **ws**: Tras ejecutar este comando accedemos a la vista de servicios web (REST). Aquí podemos ver qué servicios están ya preconfigurados y listos para ser habilitados y sus propiedades y configuración. También podemos crear nuevos servicios. Utilizaremos este comando para configurar y exponer el acceso a determinadas tablas a través del API REST, además de añadir un nuevo servicio REST para el acceso a la tabla de KPIs que crearemos.

Estudiamos también los flujos de estados (**workflows**) por los que pasan varios registros, en especial el de las incidencias, ya que necesitamos saber qué campos son necesarios para poder pasar de un estado a otro, y el nombre de los estados por los que deberemos pasar al utilizar el bot. Para comprender mejor el funcionamiento de los workflows, realizamos varias tareas con ellos, tales como modificar su flujo, añadir nuevos estados, modificar las condiciones para las transiciones etc.

Una vez que ya hemos finalizado las tareas de instalación, configuración y estudio de Service Manager, podemos comenzar el desarrollo del bot objetivo del proyecto.

8.2- Software utilizado

Para el desarrollo del proyecto se ha utilizado el siguiente software:

- **CenOs 8**: Para la máquina virtual en la que se instala el servidor de Service Manager. CentOs 8 está disponible en [2]
- **Service Manager 9.70**: Como back-end de nuestra aplicación. Nuestro sistema se comunicará con Service Manager a través de su API REST que configuraremos durante la implementación de los casos de uso. Como es software propietario de Micro Focus, no es posible añadir enlaces de descarga de la herramienta.
- **Vi**: Durante las iteraciones 1 y 2 de cada caso de uso se ha utilizado el editor vi instalado en la máquina virtual.
- **Visual Studio Code**: Para la iteración nº 3 de cada caso de uso se ha utilizado el entorno de desarrollo Visual Studio Code con su extensión para Python 3.6.8. Se ha utilizado solo en esta fase debido a la mayor facilidad para mover/editar fragmentos de código simultáneamente. Visual Studio Code está disponible en [4].
- **Python 3.6.8**: El 100% del código del proyecto está realizado en Python 3.6.8. La página oficial de Python 3.6.8 está disponible en [6]
- **Astah Professional**: Para la creación de todos los diagramas utilizados en el proyecto.
- **Microsoft Office Word**: Para la creación de este documento.
- **Microsoft Office Excel**: Para el cálculo del presupuesto

8.3- Implementación de los casos de uso

En todos los casos la implementación inicial (**iteraciones 1 y 2**) se ha hecho a partir de los diagramas realizados en los pasos previos, cumpliendo con la arquitectura propuesta, respetando el flujo de eventos, y utilizando buenas prácticas de programación, tales como la modularidad, la baja complejidad de las funciones, o la captura y tratamiento de errores en tiempo de ejecución.

Para la implementación, se han utilizado varias **librerías o APIs externas**, entre las que se encuentran el API "**Discordpy**" [7] para la comunicación del bot con el servidor de Discord y la librerías "**request**" [8] para la comunicación mediante el protocolo HTTP, o **json** [9] para la transferencia de información en formato Json.

Como anotación adicional, aunque en el caso de Python no es necesario, se ha querido evitar un acceso directo a los atributos de las clases del modelo. Para ello, se han creado métodos **gettets y setters** para los atributos necesarios (no se han incluido todos debido a que algunos atributos no se modifican u obtienen a lo largo de los casos de uso).

9- Pruebas

Para las **pruebas** y la **depuración** de cada caso de uso se ha comenzado realizando una batería de pruebas, ejecutándola posteriormente y corrigiendo cualquier error encontrado durante dichas pruebas. Durante la iteración nº 1, únicamente se ejecutan las pruebas relativas a los flujos exitosos de cada caso de uso, puesto que el objetivo de la primera iteración es conseguir que el caso de uso funcione en el escenario ideal.

9.1- Creación de nuevas incidencias en el sistema.

La batería de pruebas para el caso de uso ha sido la siguiente:

Acción	Entrada	Resultado esperado y obtenido	Relacionado con los requisitos
Solicitar creación de incidencia en el canal general.	¡newIncident	Se crea un canal y el bot informa de ello	RF001, RF008, RF009
Durante la solicitud del login del operador, enviar un nombre de operador existente	jcgd	El bot lo valida y pasa a solicitar el título de la Incidencia	RF001, RF005, RF006, RF007
Durante la solicitud del login del operador, enviar una cadena no correspondiente a ningún operador	jsqjwena	El bot informa de que no existe un operador con ese nombre y lo vuelve a pedir.	RF007
Durante la solicitud del login del operador, enviar un nombre de un contacto que no tiene operador	William	El bot informa de que no existe un operador con ese nombre y lo vuelve a pedir.	RF007
Durante la solicitud del login del operador, enviar la palabra “!exit”.	¡exit	El bot informa de que ha finalizado el caso de uso.	RF010
Durante la solicitud del login del operador, enviar la palabra “!Exit”.	¡Exit	El bot informa de que ha finalizado el caso de uso.	RF010
Durante la solicitud del login del operador, enviar la palabra “!ExIt”.	¡ExIt	El bot informa de que ha finalizado el caso de uso.	RF010
Durante la solicitud del login del operador, enviar la palabra “!EXIT”.	¡EXIT	El bot informa de que ha finalizado el caso de uso.	RF010
Durante la solicitud del título para la incidencia, introducir un título de 11 caracteres	Test title1	El bot lo valida y pasa a solicitar la descripción de la Incidencia.	RF001, RF005, RF006, RF007
Durante la solicitud del título para la incidencia, introducir un título de 10 caracteres	Test title	El bot lo valida y pasa a solicitar la descripción de la Incidencia.	RF006, RF007
Durante la solicitud del título para la incidencia, introducir un título de 9 caracteres	Not valid	El bot informa de que no es válido y vuelve a solicitarlo.	RF006, RF007

Durante la solicitud del título para la incidencia, introducir la palabra “!exit”	¡exit	El bot informa de que ha finalizado el caso de uso.	RF010
Durante la solicitud de la descripción para la incidencia, introducir una descripción de 11 caracteres	Test descri	El bot la valida y pasa a solicitar el nombre del CI afectado.	RF001, RF005, RF006, RF007
Durante la solicitud de la descripción para la incidencia, introducir una descripción de 10 caracteres	Test descr	El bot la valida y pasa a solicitar el nombre del CI afectado.	RF001, RF005, RF006, RF007
Durante la solicitud de la descripción para la incidencia, introducir una descripción de 9 caracteres	Not valid	El bot informa de que no es válida y vuelve a solicitarla.	RF006, RF007
Durante la solicitud de la descripción para la incidencia, introducir la palabra “!exit”	¡exit	El bot informa de que ha finalizado el caso de uso.	RF010
Durante la solicitud del CI afectado, escribir un CI existente en Service Manager.	Adobe Reader	El bot lo valida y pasa a solicitar un valor para el impacto de la Incidencia.	RF001, RF005, RF006, RF007
Durante la solicitud del CI afectado, escribir un CI no existente en Service Manager.	This CI doesn’t exist	El bot informa de que no existe y vuelve a solicitarlo.	RF007, RF008
Durante la solicitud del CI afectado, introducir la palabra “!exit”	¡exit	El bot informa de que ha finalizado el caso de uso y de que no se han realizado acciones.	RF010
Durante la solicitud del valor de impacto, introducir el valor “3”	3	El bot lo valida y pasa a pedir el valor de gravedad de la Incidencia.	RF001
Durante la solicitud del valor de impacto, introducir el valor “1”	1	El bot informa de que no es válido, informa de los valores posibles, y vuelve a pedir un valor.	RF007, RF008
Durante la solicitud del valor de impacto, introducir el valor “5”	5	El bot informa de que no es válido, informa de los valores posibles, y vuelve a pedir un valor.	RF007, RF008
Durante la solicitud del valor de impacto, introducir el valor “h”	H	El bot informa de que no es válido, informa de los valores posibles, y vuelve a pedir un valor.	RF007, RF008
Durante la solicitud del valor de impacto, introducir la palabra “!exit”	¡exit	El bot informa de que ha finalizado el caso de uso y de que no se han realizado acciones.	RF010
Durante la solicitud del valor de gravedad, introducir el valor “3”	3	El bot lo valida, pasa a crear y enviar la incidencia a Service Manager, e informa del fin de asistencia en el canal.	RF001, RF008
Durante la solicitud del valor de gravedad, introducir el valor “1”	1	El bot informa de que no es válida, informa de los valores posibles, y vuelve a pedir un valor.	RF007, RF008

Durante la solicitud del valor de gravedad, introducir el valor “5”	5	El bot informa de que no es válida, informa de los valores posibles, y vuelve a pedir un valor.	RF007, RF008
Durante la solicitud del valor de gravedad, introducir el valor “h”	H	El bot informa de que no es válida, informa de los valores posibles, y vuelve a pedir un valor.	RF007, RF008
Durante la solicitud del valor de gravedad, introducir la palabra “!exit”	¡exit	El bot informa de que ha finalizado el caso de uso y de que no se han realizado acciones.	RF010
Tras haber finalizado el proceso, enviar cualquier mensaje en el canal creado.	¡help	No pasa nada.	RF001
Iniciar de forma paralela varios procesos de creación de incidencias	¡newIncident (varias veces en el canal general)	Se crean varios canales para cada proceso, cada uno totalmente independiente de los demás.	RF001
Tratar de iniciar la creación de una incidencia en un canal que no sea el general	¡newIncident	Si hay un proceso en curso en el canal, se informa de que el dato introducido no es válido. Si el proceso en ese canal ya ha terminado no, no pasa nada.	RF007, RF008
Introducir datos que no sean comandos en el canal general	Adobe Reader	No pasa nada.	RF007, RF008
Introducir un comando inexistente en el canal general	¡ksjqwm	El bot informa de que no ha reconocido el comando, y proporciona la lista de los comandos disponibles.	RF008
Mientras se está ejecutando el proceso, apagar el servidor de Service Manager	<Tratar de continuar el CU>	El bot informa de que se ha perdido la conexión con el servidor y finaliza el caso de uso	RF010

Figura 33 - Batería de pruebas CU Crear Incidencia

9.2- Consulta del estado de una incidencia.

La batería de pruebas para el caso de uso ha sido la siguiente:

Acción	Entrada	Resultado esperado	Relacionado con los requisitos
Solicitar consulta de incidencia en el canal general.	¡checkIncident	Se crea un canal y el bot informa de ello	RF002, RF008, RF009
Durante la solicitud del identificador de la incidencia, enviar un identificador de incidencia existente	IM10303	El bot lo valida y pasa a contactar con Service Manager para obtener la incidencia	RF002, RF005, RF006, RF007, RF008
Durante la solicitud del identificador de la incidencia, enviar un identificador de incidencia no existente	IM99999	El bot informa de que no existe una incidencia con ese identificador y finaliza el caso de uso	RF007
Durante la solicitud del identificador de la incidencia, enviar la palabra “!exit”	¡exit	El bot informa de la finalización del caso de uso.	RF010
Durante la solicitud del identificador de la incidencia, enviar la palabra “!exit” por el canal general	¡exit	No ocurre nada en el canal creado para el caso de uso.	RF010
Mientras se está ejecutando el proceso, apagar el servidor de Service Manager	<Tratar de continuar el CU>	El bot informa de que se ha perdido la conexión con el servidor y finaliza el caso de uso	RF010

Figura 34 - Batería de pruebas CU Consultar Estado de Incidencia

9.3- Modificación de datos de una incidencia.

Acción	Entrada	Resultado esperado	Relacionado con los requisitos
Solicitar modificación de incidencia en el canal general.	¡updateIncident	Se crea un canal y el bot informa de ello	RF004, RF008, RF009
Durante la solicitud del identificador de la incidencia, enviar un identificador de incidencia existente	IM10303	El bot lo valida y pasa a solicitar los campos a actualizar	RF004, RF005, RF006, RF007
Durante la solicitud del identificador de la incidencia, enviar un identificador de incidencia no existente	IM99999	El bot informa de que no existe una incidencia con ese identificador y finaliza el caso de uso	RF007
Durante la solicitud del identificador de la incidencia, enviar la palabra “!exit”	¡exit	El bot informa de la finalización del caso de uso.	RF010
Durante la solicitud del identificador de la incidencia, enviar la palabra “!exit” por el canal general	¡exit	No ocurre nada en el canal creado para el caso de uso.	RF010

Durante la solicitud de campos a actualizar, introducir una lista válida	operator, title, description, severity, impact	Se valida la lista y se pasa a pedir los valores solicitados	RF004, RF005, RF006, RF007
Durante la solicitud de la lista de campos a modificar, enviar la palabra “!exit”	¡exit	El bot informa de la finalización del caso de uso.	RF010
Durante la solicitud de los campos de la incidencia a modificar, enviar un valor de campo no existente	operator, title,notvalid	El bot informa de que la lista no es válida y la vuelve a solicitar	RF007
Durante la solicitud de los campos de la incidencia a modificar, enviar un valor válido pero repetido	Operator, operator, title, operator	El bot valida la lista y pasa a pedir los datos, eliminando duplicados de la lista.	RF004, RF005, RF006, RF007
Durante la solicitud del login del operador, enviar un nombre de operador existente	Jcgd	El bot lo valida y pasa a solicitar siguiente campo a modificar de la Incidencia	RF004, RF005, RF006, RF007
Durante la solicitud del login del operador, enviar una cadena no correspondiente a ningún operador	Jsqjwena	El bot informa de que no existe un operador con ese nombre y lo vuelve a pedir.	RF007
Durante la solicitud del login del operador, enviar un nombre de un contacto que no tiene operador	William	El bot informa de que no existe un operador con ese nombre y lo vuelve a pedir.	RF007
Durante la solicitud del login del operador, enviar la palabra “!exit”.	¡exit	El bot informa de que ha finalizado el caso de uso.	RF010
Durante la solicitud del login del operador, enviar la palabra “!Exit”.	¡Exit	El bot informa de que ha finalizado el caso de uso.	RF010
Durante la solicitud del login del operador, enviar la palabra “!ExIt”.	¡ExIt	El bot informa de que ha finalizado el caso de uso.	RF010
Durante la solicitud del login del operador, enviar la palabra “!EXIT”.	¡EXIT	El bot informa de que ha finalizado el caso de uso.	RF010
Durante la solicitud del título para la incidencia, introducir un título de 11 caracteres	Test title1	El bot lo valida y pasa a solicitar el siguiente campo a modificar.	RF004, RF005, RF006, RF007
Durante la solicitud del título para la incidencia, introducir un título de 10 caracteres	Test title	El bot lo valida y pasa a solicitar el siguiente campo a modificar	RF006, RF007
Durante la solicitud del título para la incidencia, introducir un título de 9 caracteres	Not valid	El bot informa de que no es válido y vuelve a solicitarlo.	RF006, RF007
Durante la solicitud del título para la incidencia, introducir la palabra “!exit”	¡exit	El bot informa de que ha finalizado el caso de uso.	RF010
Durante la solicitud de la descripción para la incidencia, introducir una descripción de 11 caracteres	Test descri	El bot la valida y pasa a solicitar el valor del siguiente campo a modificar.	RF004, RF005, RF006, RF007
Durante la solicitud de la descripción para la incidencia, introducir una descripción de 10 caracteres	Test descr	El bot la valida y pasa a solicitar el valor del siguiente campo a modificar.	RF004, RF005, RF006, RF007
Durante la solicitud de la descripción para la incidencia, introducir una descripción de 9 caracteres	Not valid	El bot informa de que no es válida y vuelve a solicitarla.	RF006, RF007

Durante la solicitud de la descripción para la incidencia, introducir la palabra “!exit”	¡exit	El bot informa de que ha finalizado el caso de uso.	RF010
Durante la solicitud del valor de impacto, introducir el valor “3”	3	El bot lo valida y pasa a pedir el valor de gravedad de la Incidencia.	RF001
Durante la solicitud del valor de impacto, introducir el valor “1”	1	El bot informa de que no es válido, informa de los valores posibles, y vuelve a pedir un valor.	RF007, RF008
Durante la solicitud del valor de impacto, introducir el valor “5”	5	El bot informa de que no es válido, informa de los valores posibles, y vuelve a pedir un valor.	RF007, RF008
Durante la solicitud del valor de impacto, introducir el valor “h”	H	El bot informa de que no es válido, informa de los valores posibles, y vuelve a pedir un valor.	RF007, RF008
Durante la solicitud del valor de impacto, introducir la palabra “!exit”	¡exit	El bot informa de que ha finalizado el caso de uso y de que no se han realizado acciones.	RF010
Durante la solicitud del valor de gravedad, introducir el valor “3”	3	El bot lo valida, pasa a crear y enviar la incidencia a Service Manager, e informa del fin de asistencia en el canal.	RF004, RF008
Durante la solicitud del valor de gravedad, introducir el valor “1”	1	El bot informa de que no es válida, informa de los valores posibles, y vuelve a pedir un valor.	RF007, RF008
Durante la solicitud del valor de gravedad, introducir el valor “5”	5	El bot informa de que no es válida, informa de los valores posibles, y vuelve a pedir un valor.	RF007, RF008
Durante la solicitud del valor de gravedad, introducir el valor “h”	H	El bot informa de que no es válida, informa de los valores posibles, y vuelve a pedir un valor.	RF007, RF008
Durante la solicitud del valor de gravedad, introducir la palabra “!exit”	¡exit	El bot informa de que ha finalizado el caso de uso y de que no se han realizado acciones.	RF010
Al finalizar el proceso, señalar que se quiere continuar modificando incidencias	Yes	Comienza el proceso de nuevo	RF004, RF005, RF006, RF007
Al finalizar el proceso, señalar que no se quiere continuar modificando incidencias	No	El bot informa de que el caso de uso ha finalizado	RF004, RF005, RF006, RF007

Al finalizar el proceso, enviar el comando “!exit” o cualquier cadena distinta de “yes” o “no” (o sus variantes en mayúsculas/minúsculas)	¡exit	El bot informa de que la entrada no es válida y la vuelve a solicitar	RF010
Mientras se está ejecutando el proceso, apagar el servidor de Service Manager	<Tratar de continuar con el caso de uso>	El bot informa de que se ha perdido la conexión con el servidor y finaliza el caso de uso	RF010

Figura 35 - Batería de pruebas CU Modificar Incidencia

9.4- Cierre de una incidencia.

Acción	Entrada	Resultado esperado	Relacionado con los requisitos
Solicitar cierre de incidencia en el canal general.	¡closeIncident	Se crea un canal y el bot informa de ello	RF003, RF008, RF009
Durante la solicitud del identificador de la incidencia, enviar un identificador de incidencia existente	IM10303	El bot lo valida y pasa a solicitar la solución del problema	RF003, RF005, RF006, RF007
Durante la solicitud del identificador de la incidencia, enviar un identificador de incidencia no existente	IM99999	El bot informa de que no existe una incidencia con ese identificador y finaliza el caso de uso.	RF007
Durante la solicitud del identificador de la incidencia, enviar la palabra “!exit”	¡exit	El bot informa de la finalización del caso de uso.	RF010
Durante la solicitud del identificador de la incidencia, enviar la palabra “!exit” por el canal general	¡exit	No ocurre nada en el canal creado para el caso de uso.	RF010
Durante la solicitud de la solución de la incidencia, introducir una solución de 11 caracteres	Solution 11	El sistema lo valida y pasa a solicitar el valor del código de cierre	RF003, RF005, RF006, RF007
Durante la solicitud de la solución de la incidencia, introducir una solución de 10 caracteres	Solution10	El sistema lo valida y pasa a solicitar el valor del código de cierre	RF003, RF005, RF006, RF007
Durante la solicitud de la solución de la incidencia, introducir una solución de 9 caracteres	Solution9	El sistema informa de que no es válido y lo vuelve a solicitar	RF007
Durante la solicitud de la solución de la incidencia, enviar la palabra “!exit”	¡exit	El bot informa de que ha finalizado el caso de uso	RF010
Durante la solicitud del código de cierre de la incidencia, introducir un valor válido	Cualquier valor de 0 a 12 (incluidos)	El bot lo valida y pasa a cerrar la incidencia con los nuevos datos	RF003, RF005, RF006, RF007
Durante la solicitud del código de cierre de la incidencia, introducir un valor no válido	13	El bot informa de ello y pasa a solicitar de nuevo el valor, imprimiendo la lista de posibles valores.	RF007

Durante la solicitud del código de cierre de la incidencia, introducir el valor “!exit”	¡exit	El bot informa de que ha finalizado el caso de uso	RF010

Figura 36- Batería de pruebas CU Cerrar Incidencia

9.5- Consulta de uno o varios Indicadores Clave de Rendimiento (KPI)

Acción	Entrada	Resultado esperado	Relacionado con los requisitos
Solicitar consulta de KPIs en el canal general.	¡getKpi	Se crea un canal y el bot informa de ello	RF011, RF008, RF009
Durante la solicitud de la lista de KPIs a obtener, enviar un valor válido	1,2,3,4,5,6	El bot valida la lista, obtiene los KPIs de Service Manager y los muestra por el chat	RF011, RF008, RF009
Durante la solicitud de la lista de KPIs a obtener, enviar un asterisco	*	El bot valida la lista, obtiene todos los KPIs de Service Manager y los muestra por el chat	RF011, RF008, RF009
Durante la solicitud de la lista de KPIs a obtener, enviar una lista válida con valores repetidos	1,2,3,2,3,3,5,4,7	El bot valida la lista, obtiene todos los KPIs de Service Manager y los muestra por el chat, eliminando duplicados.	RF011, RF008, RF009
Durante la solicitud de la lista de KPIs a obtener, enviar un valor no válido	1,2,3,28	El bot informa de que algún valor no es correcto y vuelve a pedir la lista	RF007
Tras obtener la lista de KPIs, indicar que queremos continuar solicitando KPIs	yes	El bot vuelve a solicitar la lista de KPIs a obtener, comenzando el caso de uso de nuevo	RF011, RF008, RF009
Tras obtener la lista de KPIs, indicar que no queremos continuar solicitando KPIs	no	El bot informa de que ha finalizado el caso de uso	RF011, RF008, RF009
Tras obtener la lista de KPIs, introducir un valor distinto de “yes” o “no”	¡exit	El bot informa de que el valor no es válido y lo vuelve a solicitar	RF007

Figura 37 - Batería de pruebas CU Consultar KPI

10- Manual del programador

A continuación se detallan varios aspectos que podrían resultar relevantes para un futuro desarrollo de un proyecto a partir del proyecto actual:

En primer lugar, el sistema con el que se comunica el bot desarrollado (Service Manager) es un sistema **propietario** de **Micro Focus**, lo que implica que en caso de continuar el desarrollo de este proyecto será necesario obtener la licencia de dicho software. Este es otro de los motivos que ha llevado a que el proyecto se haya realizado utilizando el patrón **MVC**, ya que de esta manera si se desea desarrollar un bot de Discord para otro sistema similar, únicamente habría que modificar las llamadas al API REST en el controlador, pudiendo conservar el resto del código.

Por otro lado, se ha intentado generalizar el código lo máximo posible, para permitir una **mejor adaptación** a otras plataformas. Por este motivo, el **control de errores** se ha realizado de forma muy estricta, de tal manera que se comprueban errores que en nuestro caso no pueden llegar a ocurrir debido a la manera en que se realiza el flujo de eventos de los casos de uso, lo que proporcionaría una reducción del tiempo de desarrollo en caso de que el proyecto se ampliase en el futuro. Además, el 100% del código se ha escrito en **Python 3.6.8**, de tal manera que las comunicaciones entre las diversas clases del sistema no conllevan la necesidad de usar adaptadores u otro software especial.

Para la **comunicación con el API REST** de Service Manager se ha utilizado la librería "**Request**" [8], utilizando como formato de trasmisión de datos el formato **Json** [9], mientras que para la comunicación con Discord se ha utilizado la librería "**DiscordPy**" [7]. Los enlaces a la documentación oficial de ambas librerías se encuentran en el apartado **13-Bibliografía/Referencias** de este documento.

Para finalizar, se indica que el proyecto actual se ha realizado dentro de un escenario académico, sin intención de uso real. Es por ello que **no** se ha añadido ningún tipo de "timeout" a los procesos realizables por el usuario, lo que en un uso real podría llevar a una sobrecarga de la memoria debido a varios procesos iniciados pero no completados. Posibles soluciones a este problema serían añadir el timeout mencionado, o bien asegurarnos de que se realiza un reinicio del sistema periódicamente para evitar el posible problema.

11- Manual de usuario de las aplicaciones desarrolladas.

A continuación se detalla un manual de usuario del bot de Discord desarrollado, en el que se explicarán las posibles tareas a realizar con el bot de manera no técnica.

11.1- Obtener ayuda

Si mandamos el comando **¡help** a través del canal general, el bot nos responderá con un mensaje con las posibles acciones a realizar por parte de los usuarios.

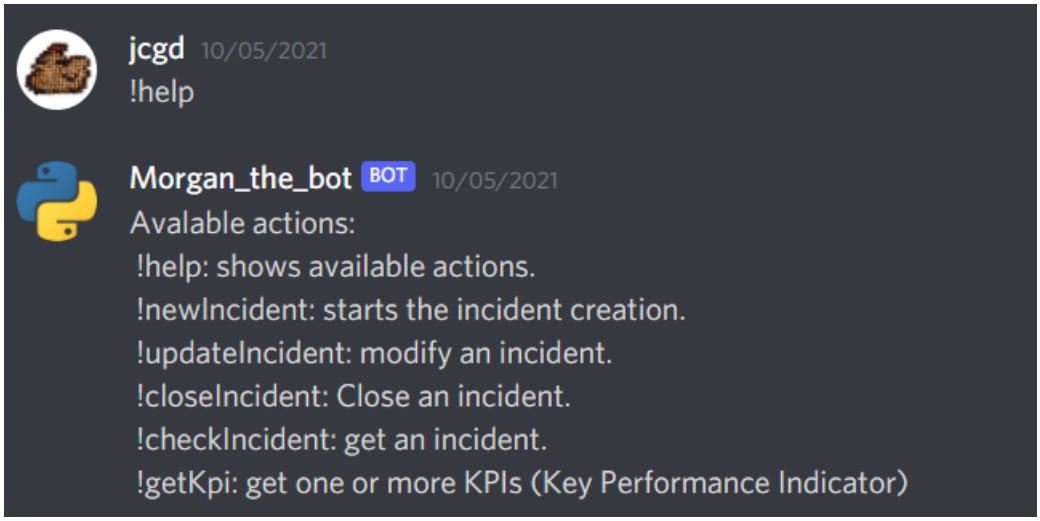

Figura 38 - Solicitar ayuda al bot de Discord

11.2- Crear Incidencia

Si enviamos el comando **¡newIncident** a través del canal general, el bot creará un nuevo canal privado exclusivo para el usuario, informando de su creación y proporcionando un enlace a dicho canal.

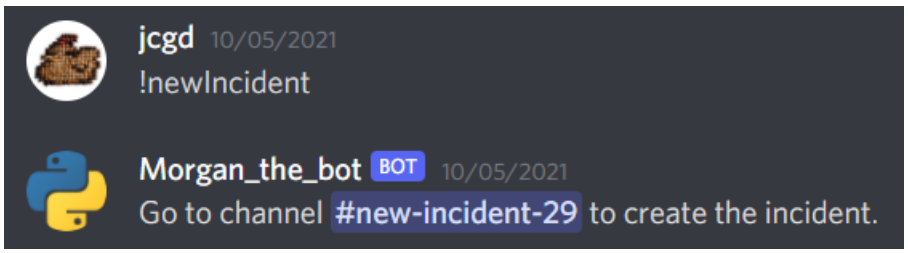

Figura 39 - Solicitar creación de nueva Incidencia

En el nuevo canal se irán solicitando los datos de manera guiada, validando los datos según los introduce el usuario, de tal manera que se evita continuar con el caso de uso hasta no asegurar que los datos son correctos.

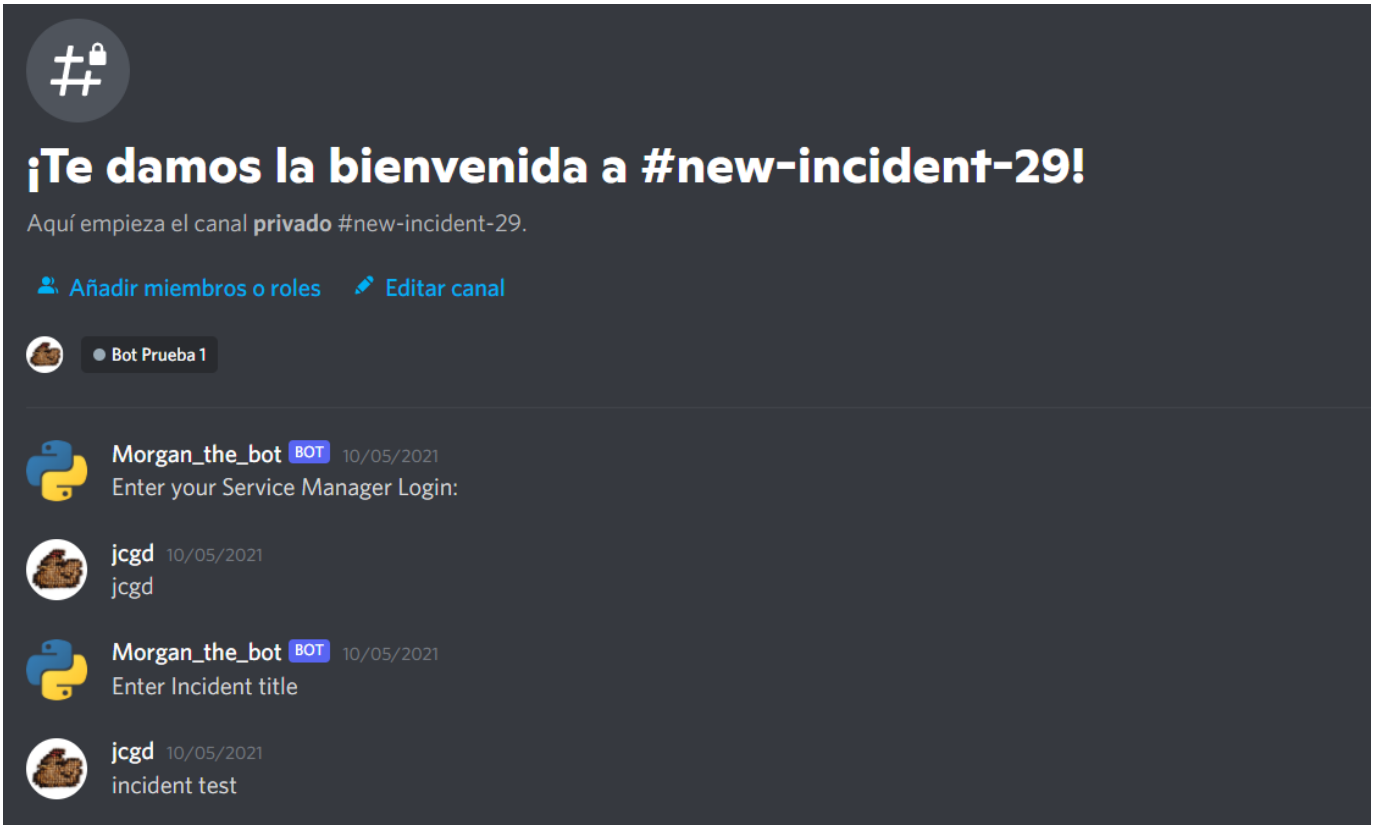

Figura 40 - Canal de Creacion de la Incidencia

11.3- Modificar Incidencia

Si enviamos el comando **¡updateIncident** a través del canal general, el bot creará un nuevo canal privado exclusivo para el usuario, informando de su creación y proporcionando un enlace a dicho canal.

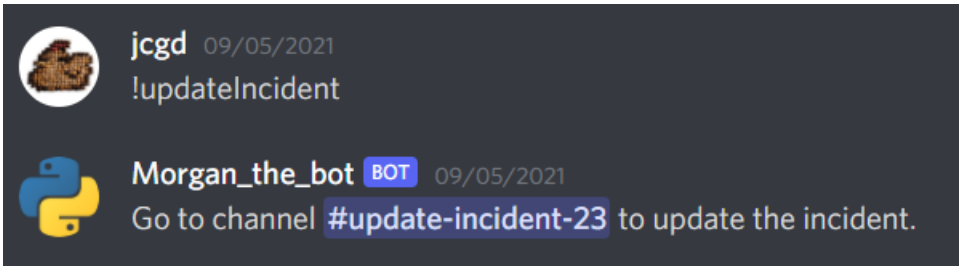

Figura 41 - Solicitar modificación de una Incidencia

En el nuevo canal se irán solicitando los datos de manera guiada, validando los datos según los introduce el usuario, de tal manera que se evita continuar con el caso de uso hasta no asegurar que los datos son correctos.

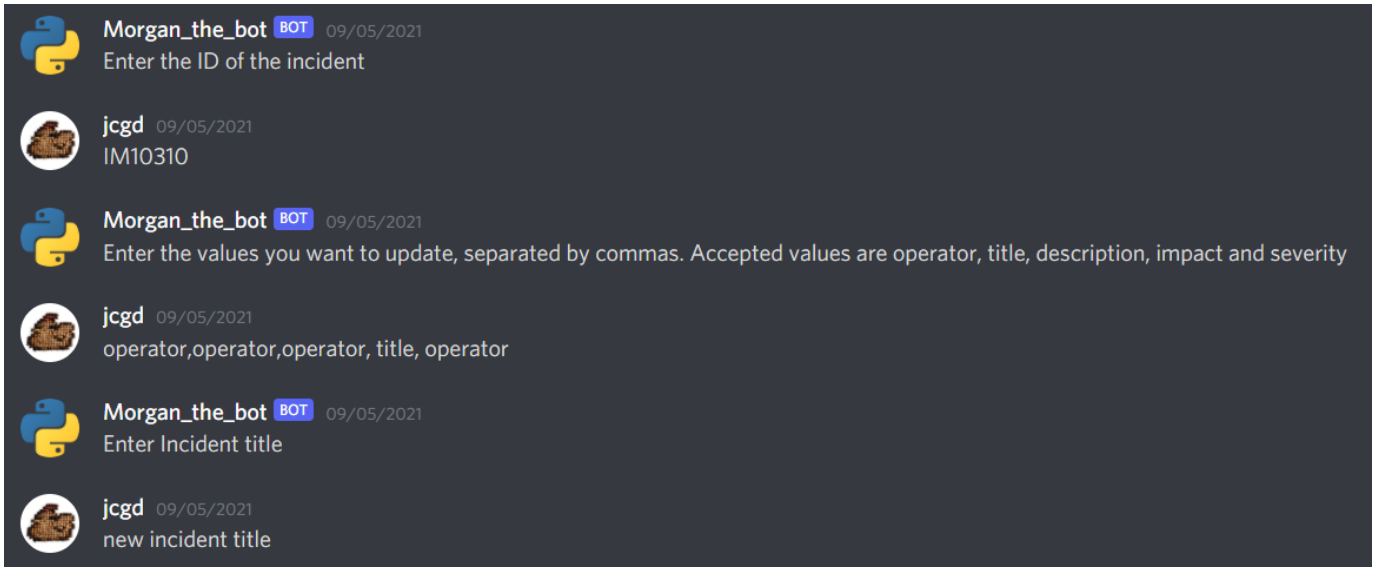

Figura 42 - Canal de modificación de la Incidencia

No se expondrán el resto de casos de uso debido a que el proceso es similar a los descritos anteriormente. **En todos los casos los procesos se iniciarán a través del canal general**, lo que provocará la creación de un canal privado en el que el usuario pueda continuar el proceso de manera confidencial.

12- Conclusiones

Para finalizar con el documento, se resumirán las **características del sistema desarrollado**, así como los **desafíos** encontrados a lo largo del proyecto, y las posibles **continuaciones** que podría tener el proyecto.

12.1- Características del proyecto

- Se ha instalado y configurado una máquina virtual CentOs 8, incluyendo aspectos de cortafuegos, usuarios, bases de datos, servidores web e instalación de diversas aplicaciones mediante la terminal.
- Se ha instalado y desplegado un servidor de Service Manager, y se ha aprendido a realizar acciones básicas de “customización” de la aplicación.
- Se han creado diagramas de análisis y de diseño de la aplicación, con objetivo de tener un mayor conocimiento del sistema antes de desarrollarlo.
- Se ha realizado una estimación de riesgos, costes y duración del proyecto para planificar correctamente el desarrollo del proyecto.
- Se ha creado un bot en Python que se comunica con Service Manager a través de su API REST, y con el servidor de Discord definido en el fichero de configuración .env.
- El bot diseñado se ha implementado aplicando el patrón MVC, obteniendo así un código fácilmente reutilizable.
- Se ha hecho especial hincapié en la robustez del código para evitar fallos, añadiendo incluso comprobación redundante de errores para asegurar un correcto funcionamiento en caso de que el proyecto se amplíe en el futuro.
- Se ha realizado una revisión de la calidad del código con objetivo de obtener métodos robustos, de un tamaño aceptable, y de baja complejidad.
- Para finalizar, se ha realizado el presente documento para tratar explicar el proceso seguido a lo largo del proyecto.

12.2- Futuras aplicaciones

Entre las futuras aplicaciones que podría tener el proyecto, podría incluirse la de una **integración real con Service Manager**, permitiendo así a los usuarios una tercera forma de contacto con el servidor aparte de los dos clientes disponibles actualmente, o una **ampliación** de los servicios ofrecidos por el bot, aumentando su funcionalidad.

Por otro lado, la modularidad del código permite su **fácil adaptación a otras plataformas** de chat, tales como Telegram, Facebook o Twitter, aumentando las posibles opciones de acceso a Service Manager.

13- Bibliografía/Referencias

[1] Repositorio del proyecto: https://github.com/jcgd2796/TFG_BotDiscord (fecha de última revisión: 14/06/2021).

[2] Página oficial de CentOs: https://www.centos.org/download (fecha de última revisión: 12/05/2021)

[3] Guía de instalación de PostgreSQL 12 en CentOs 8: https://www.digitalocean.com/community/tutorials/how-to-install-and-use-postgresql-on-centos-8-es (fecha de última revisión: 13/05/2021)

[4] Página oficial de Visual Studio Code: https://code.visualstudio.com/ (fecha de última revisión: 13/05/2021)

[5] Guía de instalación de Service Manager: https://docs.microfocus.com/itom/Service_Manager:9.70/InstallSMServer (fecha de última revisión: 13/05/2021)

[6] Página principal de Python 3.6.8: https://www.python.org/downloads/release/python-368/ (fecha de última revisión: 13/05/2021)

[7] Documentación del API DiscordPy: https://discordpy.readthedocs.io/en/latest/api.html# (fecha de última revisión: 13/05/2021)

[8] Guía de uso de la librería request: https://realpython.com/python-requests/ (fecha de última revisión: 13/05/2021)

[9] Guía de uso de la librería json: https://docs.python.org/3/library/json.html (fecha de última revisión: 13/05/2021)