Universidad de Oviedo

Escuela de Ingeniería Informática

GestIncis: Sistema de Gestión de Incidencias

Arquitectura Software para GestIncis.
Descripción del trabajo práctico (2018)

Descripción de la práctica de Arquitectura del Software durante el curso 2017-18.

Escuela de Ingeniería Informática,
Univ. Oviedo

07 de mayo de 2018

GRADO DE INGENIERÍA INFORMÁTICA DEL SOFTWARE

GestIncis: Sistema de Gestión de Incidencias

Autores: **Jesús García Minas.**

Pelayo García Torre.

José Antonio García García.

César Camblor García.

Pablo Díaz Rancaño.

Fernando De la Torre Cueva.

Pablo Álvarez Álvarez.

Fecha: 07 de mayo de 2018

Versión: Final

Tabla de contenido

1 Introducción y Objetivos

El objetivo de este documento es definir la estructura de una arquitectura que se encargará de gestionar una variedad de incidencias con la finalidad de solucionarlas. El sistema aquí descrito tiene como funcionalidad principal gestionar avisos por parte de diferentes agentes, que se listarán más adelante, y mejorar la eficiencia en su resolución.

La arquitectura aquí descrita forma parte del primer entregable de la asignatura Arquitectura del Software, impartida en el grado de Ingeniería Informática del Software, Escuela de Ingeniería Informática, Universidad de Oviedo por los autores de este documento.

El sistema se ha descompuesto en dos módulos iniciales:

- Loader: módulo encargado de cargar datos de los agentes participantes en el sistema para su utilización.

- Agents: módulo encargado de mostrar información sobre los agentes del sistema vía Web.

Estos dos primeros módulos han sido implementados en dos semanas de trabajo por los autores de este documento utilizando el proyecto base de los alumnos de esta misma asignatura en el curso 2016 - 2017 que figuran en esta lista: Sergio Flórez Vallina, Rubén García Ruiz, Sonia Gestal Huelga, Luis Irazusta Lorenzo, Daniel Bermejo Blanco, Pedro Fernández Álvarez, Darío Alonso Díaz, Jonathan Vallés Robla.

Posteriormente se han creado dos nuevos módulos:

- InciManager: módulo encargado de tramitar incidencias enviadas por los agentes.

- InciDashboard: módulo encargado de gestionar las incidencias a través de un cuadro de mandos.

2 Requisitos

2.1 Loader

Este módulo se encarga de cargar los datos de los agentes que podrán enviar incidencias al sistema. A diferencia del curso pasado, hay diferentes tipos de agentes: personas físicas, entidades, sensores, etc. Cada tipo de usuario estará identificado por una palabra clave. Ejemplos de identificadores del tipo de agente son: "Person", "Entity", "Sensor", etc.

La introducción de los datos se hará a partir de ficheros Excel. Los campos del fichero Excel son diferentes a los del sistema anterior, los campos son:

- Nombre (en el caso de personas, contendrá nombre y apellidos)
- Localización (coordenadas geográficas del agente). Este valor es opcional para personas y entidades. Si no hay localización el valor estará en blanco.
- Email: Correo electrónico de contacto. En el caso de sensores u otro tipo de agentes automáticos, puede ser el correo electrónico de la persona que lo administra.
- Identificador: Identificador del agente. En caso de personas físicas o entidades puede ser el CIF. Este identificador será único en el sistema y será el nombre de usuario.
- Tipo: Número entero que representa el tipo de agente.

Además del fichero Excel que contiene los datos de cada agente, el sistema utilizará un fichero maestro en formato CSV que contiene los tipos de agentes disponibles. El fichero tiene 2 campos separados por comas donde el primer campo es el código numérico y el segundo el nombre del tipo de usuario. Por ejemplo:

1,Person

2,Entity

3,Sensor

Los códigos y tipos de agentes que aparezcan en este fichero maestro pueden variar, y el sistema de carga tendrá en cuenta los códigos numéricos para resolver el tipo de entidad que corresponda.

Este módulo analizará los datos de los agentes, creando un informe de errores, si se producen. Por cada agente, se almacenará la información proporcionada, junto con una clave de acceso que se generará aleatoriamente. Toda esa información será almacenada en una base de datos que será utilizada por el otro módulo.

Además, se creará una lista de cartas personalizadas para informar del usuario y clave introducido en el sistema y que se enviarán a los correos electrónicos que se han indicado. A diferencia del año pasado, el usuario del sistema será el campo identificador (el año pasado era el correo electrónico).

El sistema podría extenderse para emitir cartas en formatos como Word o PDF comunicándole a la persona su nombre de usuario y su clave de acceso.

Si el fichero viniera con errores, se detectarían y se enviarían los datos a un fichero de LOG para su

posterior tratamiento.

El analizador de los datos de entrada debe ser configurable, ya que podrían venir los datos en diferentes formatos y no sólo en Excel. Es opcional permitir más de una entrada, pero es obligatorio que el sistema permita en el futuro una ampliación de manera sencilla.

2.2 Agents

Al igual que en el curso pasado, este módulo consiste en un servicio Web REST que permitirá consultar y obtener información de los agentes que participan en el sistema. El servicio Web está pensado para ser utilizado por agentes que puedan conectarse al sistema, por lo que la entrada serán ficheros en formato JSON y las respuestas también serán en formato JSON.

Al igual que el año pasado, este sistema podrá disponer de un subsistema de acceso a través de Web para actualizar la clave de cada usuario. Opcionalmente, podrán actualizarse otros campos de los usuarios. El formato de las invocaciones al sistema es el siguiente (obsérvese que se han modificado 2 campos respecto al año pasado):

```
{"login": usuario, "password": password, "kind": tipo de agente}
```

En caso de que la combinación login/password/kind aparezca en la base de datos, se devolverá la siguiente información:

```
{
        "id": "id_usuario" (long),
        "password": "password",
        "kind": "tipo_usuario",
        "kindCode": "id_kind" (long),
        "dni": "dni",
        "nombre": "nombre",
        "apellidos": "apellidos",
        "email": "email",
        "username": "nombre_usuario"
}
```

El campo "kindCode" se obtiene a partir de un fichero maestro en formato CSV que se ha descrito en la sección anterior.

Se puede crear un sencillo interfaz de acceso en HTML para que los usuarios puedan entrar en el sistema, consultar su información o incluso modificar algunos datos.

2.3 InciManager

Este módulo se encarga de tramitar las incidencias que serán enviadas por los agentes. El módulo se comunicará con el módulo 2 (Agents) para chequear que los agentes están dados de alta en el sistema y tienen permisos para enviar incidencias.

Pueden existir diferentes tipos de agentes para enviar incidencias: personas, entidades, sensores, etc. El tipo de agentes no está predeterminado, sino que se obtendrá del mismo fichero maestro descrito en el primer entregable. Cada incidencia puede contener los siguientes campos: nombre de usuario y password, nombre de la incidencia, descripción, localización (se obtendrá automáticamente del dispositivo si es posible), etiquetas (lista de palabras separadas por comas que permitirán categorizar las incidencias), información adicional (fotos, vídeos, etc.).

Algunas incidencias podrán también contener una lista de campos con la forma "propiedad/valor", donde el campo propiedad indica un nombre de propiedad, y el campo valor, indica el valor de dicha propiedad.

En caso de que la combinación "nombre de usuario/password" sea incorrecta, las incidencias no serán procesadas y se informará del error.

Si la combinación es correcta, se procesarán las incidencias y se enviará la incidencia procesada a Apache Kafka para su posterior análisis por parte del módulo 4 (InciDashboard).

Los sensores pueden estar enviando incidencias, de forma continua, dependiendo de cómo estén configurados. Por ejemplo, un sensor de temperatura puede estar configurado para emitir incidencias cada hora con información sobre la temperatura en un determinado lugar.

Las incidencias adquirirán un estado (abierta, en proceso, cerrada, anulada) así como otra información generada por el sistema (persona/entidad asignada para resolver la incidencia), comentarios sobre la incidencia realizados por los operarios, etc. Las incidencias también pueden tener una caducidad, por ejemplo, en el caso de las incidencias que pueda enviar un sensor de temperatura, si se envían cada hora, tendrán una hora de caducidad). El sistema solamente almacenará un subconjunto de las incidencias enviadas (por ejemplo, las enviadas por personas o entidades, o algunos valores específicos enviados por los sensores). Las cuales podrán consultar las incidencias que han enviado utilizando el nombre de usuario y la clave asignadas en el módulo 1. Cada agente podrá tener acceso a todas las incidencias que ha enviado y realizar un seguimiento de estas.

2.4 InciDashboard

El cuadro de mandos está pensado para que sea utilizado por el personal de gestión de incidencias, que podrá visualizar y gestionar las incidencias que ocurren en el sistema.

Este módulo recibirá las incidencias suministradas por el módulo 3 (InciManager) a través de Apache Kafka.

Ciertos agentes, como los sensores, estarán enviando continuamente incidencias, con los valores de ciertas propiedades. El cuadro de mandos se configurará indicando qué valores pueden permitirse en ciertas propiedades y cómo clasificar los valores de otras propiedades.

En caso de que los valores de ciertas propiedades sean peligrosos, el sistema notificará a los operarios del sistema para que tomen las acciones correspondientes.

El sistema ofrecerá una monitorización continua de la evolución de los valores de las propiedades más representativas de los sensores, así como de las incidencias que estén siendo generadas por las personas o entidades.

También se ofrecerá la posibilidad de visualizar las incidencias geolocalizadas en un mapa, así como los valores actuales, los estados y los históricos de algunas (por ejemplo, de la temperatura).

El sistema ofrecerá información a los operarios sobre las incidencias que tienen asignadas y les permitirá controlar dichas incidencias, cambiando el estado de las incidencias según se hayan procesado o no.

El sistema podrá ofrecer diferentes visualizaciones gráficas o estadísticas de las incidencias.

3 Metodología usada

Se va a realizar un estudio de arquitectura siguiendo el método de ADD (Bass, Clements, & Kazman, 2003) y la norma del SEI (ANSI/IEEE 1471, 2000).

La documentación sigue el esquema propuesto en la guía de aprendizaje de la asignatura y también se han tomado algunas secciones siguiendo las plantillas propuestas en arc42 (http://arc42.org/). Las plantillas actuales están en inglés y alemán, pero había una versión anterior con plantillas en español.

Existe un proyecto que usa esas plantillas para documentar una arquitectura de software sencilla sobre una aplicación de gestión de rutas de bicicleta. La documentación está disponible en la Web. Se puede ver aquí:

http://biking.michael-simons.eu/docs/index.html

4 Identificación de Stakeholders

En este caso los *stakeholders* (personas interesadas) son:

1. Desarrolladores equipo e2a1
2. Desarrolladores equipo e2a2
3. Administrador del Sistema
4. Agentes
5. Operarios
6. Profesores de la asignatura

Así pues, la lista de *stakeholders* queda:

Código	Stakeholder	Intereses (Módulos)
ST-01	Desarrolladores equipo e2a1	Los 4 submódulos
ST-02	Desarrolladores equipo e2a2	Los 4 submódulos
ST-03	Administrador de sistemas	Carga de ficheros
ST-04	Agentes	Creación de avisos
ST-05	Operarios	Comprobaciones de los datos, trámite de incidencias y cuadro de mandos
ST-06	Profesores de la asignatura	Los 4 submódulos

Tabla 1. Lista de Stakeholders e intereses

Posteriormente se pasa a describir en más detalle cada uno.

4.1 Desarrolladores equipo e2a1.

Se trata del equipo de desarrollo e2a1.

Entre sus objetivos están:

- Utilizar tecnologías y metodologías conocidas, minimizando los riesgos relacionados con el aprendizaje de las nuevas.
- Aprender técnicas de desarrollo de software de forma colaborativa y profesional.
- Utilización de tecnologías similares a las del grupo e2a2 con quien deberán integrarse posteriormente para evitar incompatibilidades.
- Implementación del módulo de carga de ficheros a la base de datos (Loader).
- Documentación del módulo encargado de tramitar las incidencias (InciManager).
- Implementación del módulo encargado de visualizar y gestionar las incidencias en un cuadro de mandos (InciDashboard).

4.2 Desarrolladores equipo e2a2.

Se trata del equipo de desarrollo e2a2.

Entre sus objetivos están:

- Utilizar tecnologías y metodologías conocidas, minimizando los riesgos relacionados con el aprendizaje de las nuevas.
- Aprender técnicas de desarrollo de software de forma colaborativa y profesional.
- Utilización de tecnologías similares a las del grupo con quien deberán integrarse posteriormente para evitar incompatibilidades.
- Implementación del módulo encargado de consultar los datos de los agentes del sistema (Agents).
- Documentación del módulo encargado de visualizar y gestionar las incidencias en un cuadro de mandos (InciDashboard).
- Implementación del módulo encargado de tramitar las incidencias (InciManager).

4.3 Administrador del Sistema

Es la persona que carga los ficheros de datos. Entre sus objetivos están:

- Tecnologías sencillas de los ficheros de entrada.
- Ficheros que puedan leerse por los humanos.
- Ser capaz de automatizar el proceso de carga de listas de agentes.
- Ser capaz de depurar el proceso de carga en caso de errores.

4.4 Agentes

Son un grupo de los usuarios finales. Están formados por ciudadanos, entidades, sensores, etc.

Entre sus objetivos están:

- Sencillez de acceso a los datos.
- Ser capaz de acceder desde sus respectivos puntos de acceso de una forma segura.
- Ser capaz de enviar incidencias al sistema.
- Ser capaz de consultar el historial de sus incidencias.

4.5 Operarios

Son el otro grupo de usuarios finales. Está formado por el equipo que se encarga de recoger los datos proporcionados para los agentes y tratar de solucionarlos.
Entre sus objetivos están:

- Disponer de los datos de las incidencias tan pronto como sea posible y resolverlas.
- Consultar el tipo de agente que ha registrado la incidencia para comprobar su veracidad.
- Visualizar la lista de incidencias que hay en el sistema.
- Utilizar tecnologías fáciles de usar e interoperables con otros sistemas.
- Realizar comentarios sobre las incidencias que tienen asignadas.

4.6 Profesores de la asignatura

Son los responsables de los resultados de la práctica.

Entre sus objetivos están:

- Proponer tecnologías que ayuden a los estudiantes a adquirir habilidades relacionadas con la arquitectura del software mediante el desarrollo de un proyecto práctico.
- Introducir a los estudiantes en el desarrollo de software de forma colaborativa y profesional, mediante desarrollo basado en pruebas (TDD, *test-drivendesign*)
- Proponer un trabajo de desarrollo a partir de una documentación que pueda realizarse en el tiempo asignado por los estudiantes de la asignatura
- Mostrar a los estudiantes un ejemplo de documentación de arquitectura.

5 Atributos de calidad

Para el sistema se han identificado los siguientes atributos de calidad: Lista de atributos de calidad

Código	Descripción	Tipo de Atributo	Módulo afectado
AT001	Disponibilidad del sistema 364/365	Disponibilidad	Agents, InciDashboard e InciManager
AT002	Facilidad de cambio de diversas partes de la aplicación: Cambiar el parser de entrada de listas de agents.	Modificabilidad	Agents
AT003	Facilidad de cambio de diversas partes de la aplicación: Añadir nueva información al fichero de *log*	Modificabilidad	Loader
AT004	Facilidad para modificar partes de la aplicación: Añadir otros formatos de salida o de entrada	Modificabilidad	Loader
AT005	BORRADO		
AT006	Facilidad para cambiar partes de la aplicación: procesar y devolver información en otros formatos mediante negociación de contenido. Posibilidad de cambiar de JSON a otro estándar.	Modificabilidad	Agents
AT007	El rendimiento del proceso de carga de datos de los ficheros es razonable (no demasiado lento, pero tampoco crítico)	Rendimiento	Loader
AT008	El sistema debe garantizar la confidencialidad de los datos de los usuarios	Seguridad	Agents, Loader, InciManager, InciDashboard
AT009	Debe ser posible chequear que el servicio web se comporta adecuadamente	Testabilidad	Agents
AT010	Debe ser posible chequear el comportamiento del sistema de carga de datos	Testabilidad	Loader
AT011	El sistema debe poder ser ejecutado por técnicos de sistemas familiarizados con herramientas tipo Unix	Usabilidad	Loader, Agents, InciDashboard e InciManager
AT012	El servicio REST debe poder ser utilizado por procesos automáticos que consulten el estado de un usuario	Interoperabilidad	Agents
AT013	El sistema debe ser sencillo y fácil de implementar	Simplicidad	Agents, Loader, InciDashboard e InciManager
AT014	El sistema debe ser fácilmente desplegable	Desplegabilidad	Agents, InciDashboard e InciManager
AT015	El sistema debe denegar el acceso a agentes que no se encuentren registrados.	Seguridad	InciManager, Agents

Código	Descripción	Tipo de Atributo	Módulo afectado
AT016	El sistema debe enviar y procesar las incidencias lo antes posible.	Rendimiento	InciManager y InciDashboard
AT017	BORRADO		
AT018	BORRADO		
AT019	El sistema debe permitir visualizar gráficamente las incidencias.	Usabilidad	InciDashboard
AT020	BORRADO		
AT021	El sistema debe poder guardar la fecha de llegadas de las incidencias para futuras gestiones.	Persistencia	InciManager e InciDashboard
AT022	Los agentes deben poder visualizar su historial de incidencias	Disponibilidad	InciManager
AT023	Todos los datos del sistema serán guardados en una base de datos relacional.	Persistencia	Loader, InciDashboard
AT024	Los operarios deben poder visualizar sus incidencias asignadas y gestionarlas	Disponibilidad	InciDashboard
AT025	El uso del gestor de incidencias tendrá una curva de aprendizaje leve.	Usabilidad	InciDashboard

Tabla 2. Lista de atributos de calidad y tipos

5.1 Atributos de calidad

Los diferentes atributos de calidad son de interés para alguno de los *stakeholders*. La siguiente tabla muestra la lista de intereses para el proyecto actual:

Atributos vs Interesados	ST-01	ST-02	ST-03	ST-04	ST-05	ST-06
AT001	X	X	X	X	X	X
AT002	X	X				X
AT003	X	X				X
AT004	X	X	X			X
AT005	-	-	-	-	-	-
AT006	X	X				X
AT007	X	X	X			X
AT008	X	X	X	X	X	X
AT009	X	X	X	X		X
AT010	X	X	X			X
AT011	X	X	X			X
AT012	X	X				X
AT013	X	X	X			X
AT014	X	X	X			X
AT015	X	X		X		X
AT016	X	X		X	X	X
AT017	-	-	-	-	-	-
AT018	-	-	-	-	-	-
AT019	X	X			X	X
AT020	-	-	-	-	-	-
AT021	X	X			X	X
AT022	X	X		X		X
AT023	X	X	X	X	X	X
AT024	X	X			X	X
AT025	X	X			X	X

Tabla 3. Lista de intereses de los *stakeholders*

6 Restricciones

Para realizar esta aplicación existen las siguientes restricciones

6.1 Restricciones técnicas

Código	Restricción	Motivación
TC001	El lenguaje de programación será Java	Es el lenguaje dominado por el equipo de desarrollo (ST001 y ST002)
TC002	Se utilizará una base de datos relacional para almacenar los datos	El equipo de desarrollo (ST001 y ST002) tiene conocimientos de bases de datos relacionales y existen múltiples librerías para trabajar con bases de datos relacionales desde Java
TC003	El servicio Web estará basado en estilo REST con formato de entrada JSON	El estilo REST es fácil de implementar y consumir. Y JSON es uno de los lenguajes más utilizados, por lo que es soportado por una gran variedad de sistemas.
TC004	Los datos de entrada vienen en formato Excel	Excel es un formato de datos bastante popular y existen varias librerías Java para procesar ficheros Excel.
TC005	El formato de salida de los emails personalizados será texto plano	Con el fin de facilitar la implementación se propone generar cartas personalizadas mediante texto plano. El equipo de desarrollo puede opcionalmente implementar otros formatos
TC006	Pruebas automáticas y desarrollo basado en pruebas	Las pruebas deberán ser ejecutables automáticamente. Se propone un desarrollo basado en pruebas, así como la utilización de técnicas de integración continua.
TC007	El servicio Web se implementará mediante el *framework* Spring Boot	El framework Spring Boot se basa en Spring, que es un *framework* Java muy popular en la industria. Existen muchos ejemplos y material de ayuda para facilitar el aprendizaje por parte de los estudiantes.
TC008	Se utilizará Apache Kafka para recibir y procesar las incidencias realizadas por los agentes	Apache Kafka es un proyecto de intermediación de mensajes escrito en Scala que ofrece un alto rendimiento junto a una baja latencia para la manipulación en tiempo real de fuentes de datos.

Tabla 4. Restricciones técnicas

6.2 Restricciones organizativas

Código	Restricción	Motivación
OC001	Cada sub-sistema será implementado por un equipo pequeño de estudiantes.	El tamaño de los equipos será de unos 3 ó 4 estudiantes con el fin de que los estudiantes puedan aprender a desarrollar software de forma colaborativa mediante un proyecto simple.
OC002	La estructura de la base de datos será la misma para los 2 sub-sistemas junto a los nuevos sub-sistemas añadidos.	El pegamento entre los sub-sistemas es la base de datos, cuya estructura debe ser acordada por los 2 equipos. Dado el pequeño tamaño de estos, separarlos en varias bases de datos no aportaría ninguna ventaja respecto a utilizar una conjunta
OC003	El código fuente será gestionado mediante el sistema control de versiones Git en un repositorio público en github	Los sistemas de control de versiones son utilizados por la mayoría de las empresas de desarrollo de software. Github ofrece un software de gestión de proyectos muy potente

Tabla 5. Restricciones organizativas

6.3 Restricciones

Los diferentes atributos de calidad son de interés para alguno de los *stakeholders*. La siguiente tabla muestra la lista de intereses para el proyecto actual:

Atributos vs Interesados	ST-01	ST-02	ST-03	ST-04	ST-05	ST-06
TC001	X	X				X
TC002	X	X				X
TC003	X	X				X
TC004	X	X	X			X
TC005	X	X		X		X
TC006	X	X				X
TC007	X	X				X
TC008	X	X				X
OC001	X	X				X
OC002	X	X				X
OC003	X	X				X

7 Ámbito del sistema y contexto

Para describir la solución se utilizarán diagramas contextuales y texto.

La aplicación está partida en cuatro procesos:

- Loader: Se encarga de la carga de los ficheros. Utiliza el estilo Batch.
- Agents: Se encarga de las comprobaciones de los agentes. Utiliza el estilo Call-Return.
- InciManager: Se encarga de tramitar las incidencias, enviadas por agentes con permisos. Utiliza el estilo MVC.
- InciDashboard: Se encarga de evaluar las incidencias en tiempo real y avisar de un posible problema. Utiliza el estilo MVC.

Todos se integran usando el estilo arquitectónico de datos compartidos.

En el diagrama de contexto de la Figura 1. Contexto de negocio del sistema, se muestran las principales interfaces de cada sistema. El subsistema Database es común a los módulos Agents y Loader, por tanto, hay que acordar la tecnología, el modelo de base de datos y el modo de acceso. Por otro lado, el módulo InciDashboard, encargado de gestionar las incidencias enviadas por el módulo InciManager.

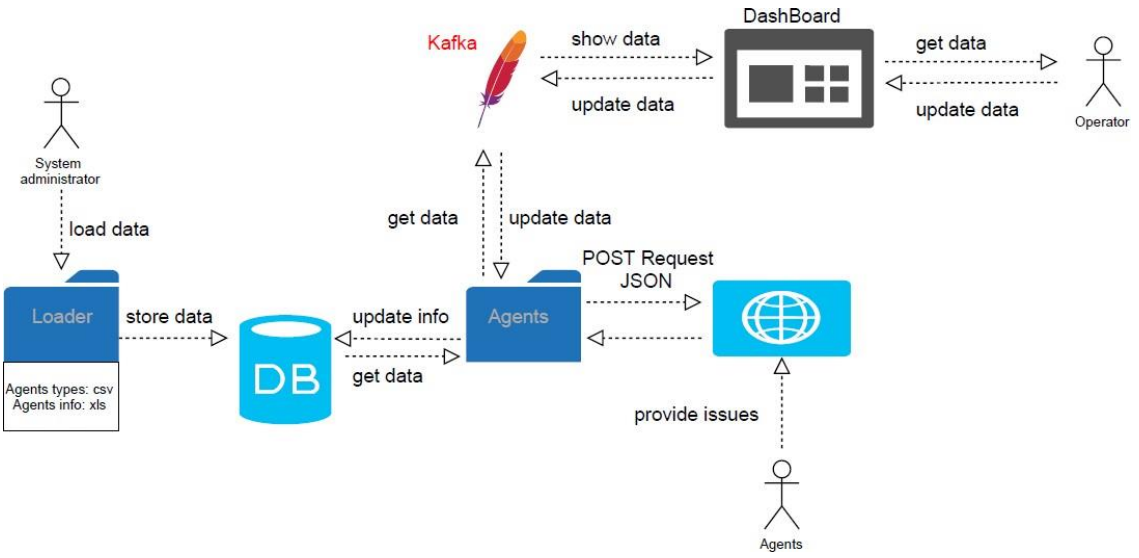


Figura 1. Contexto de negocio del sistema

A continuación, se incluye un diagrama BPMN que define el proceso completo de ambos subsistemas.
A destacar:

- Hay una base de datos común.
- Los datos intercambiados usan formato JSON.
- Los procesos de Loader y de Agents son asíncronos.
- Apache Kafka se encarga de almacenar y transmitir las incidencias.

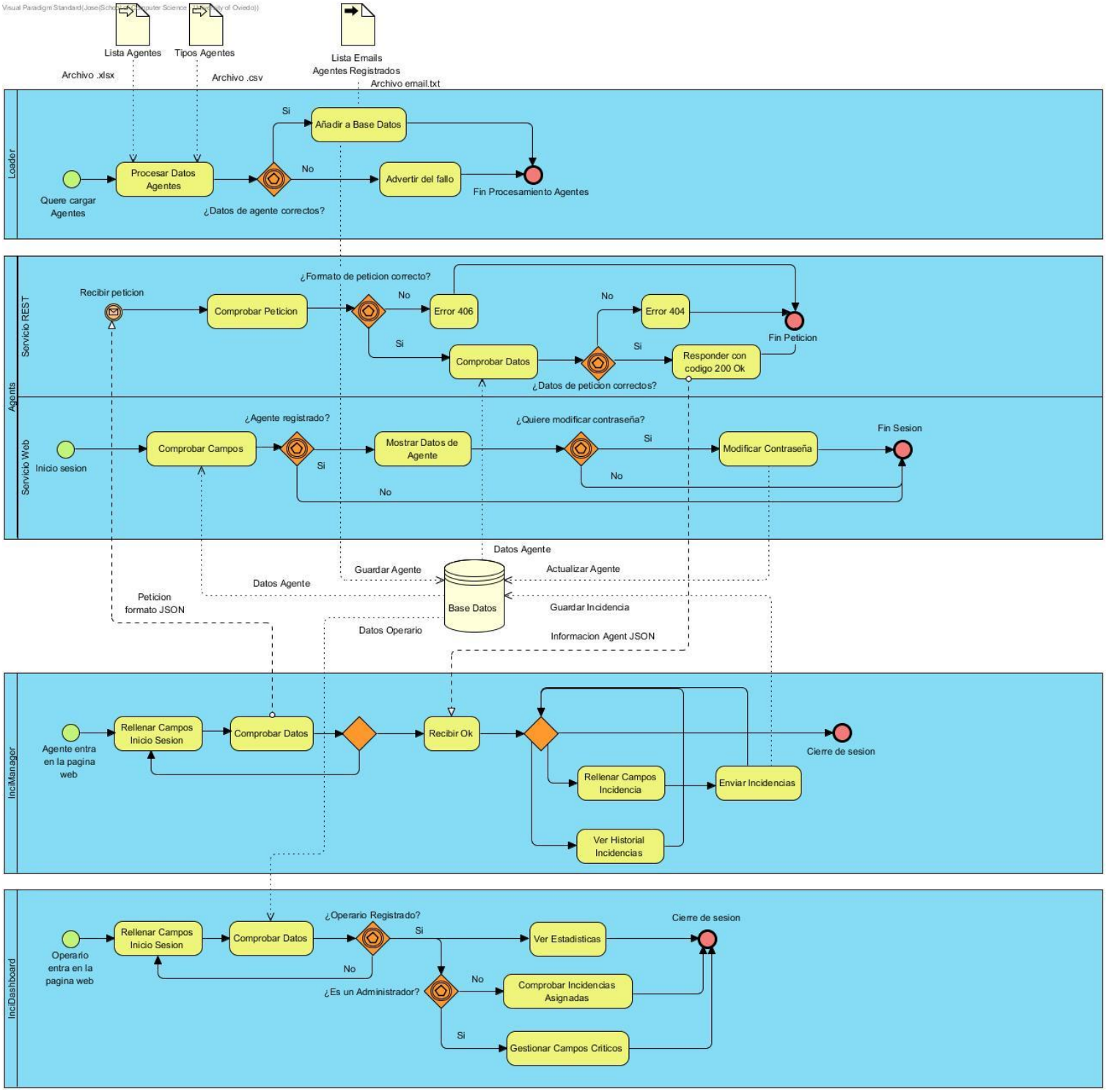


Figura 2. Diagrama BPMN

8 Escenarios de calidad

Con toda la información anterior se procederá a definir los escenarios de calidad que influencian esta arquitectura.

En las próximas páginas se muestra una tabla con la lista de escenarios identificados.

Escenario Nº	Fuente de estímulo	Estímulo	Entorno	Artefacto	Respuesta	Medición de la respuesta	Atributo de calidad afectado
1	Sistema	Realizar cualquier acción sobre el sistema	Despliegue	Agents, InciManager, InciDashboard	El sistema recibe la respuesta adecuada	La información es recibida en menos de 20 seg. A cualquier hora del día.	**AT001**
2	Desarrollador	Se introduce un nuevo Parser	Desarrollo	Parser	La modificación es introducida adecuadamente	El sistema es compilado y pasa todas las pruebas	**AT002**
3	Desarrollador	Se implementan nuevos registros para la generación de informes de error	Desarrollo	ReportWriter, DBUpdate y Parser	La opción es implementada con cambios mínimos que solamente afectan al módulo de generación de informes	Menos de un día de trabajo.	**AT003**
4	Desarrollador	Se añade un nuevo formato de salida	Desarrollo	Loader y DBManagement	Se incluye el nuevo formato con cambios mínimos en el código	Menos de un día de trabajo	**AT004**
5	BORRADO						**AT005**
6	Desarrollador	Se añade un nuevo formato a los web- services	Desarrollo	Agents	Se implementa el nuevo formato	Menos de dos días de trabajo	**AT006**
7	Administrador del Sistema	Cargar una hoja excel en el sistema (DB)	Tiempo de ejecución	Parser, DBUpdate and ReportWriter	Se carga una Excel sin errores en un tiempo razonable	< 1 segundo por cada 10 agentes	**AT007**
8	Desarrollador	Cargar un sistema en el sistema (DB)	Desarrollo/ Tiempo de ejecución	Parser, DBUpdate and ReportWriter (Optional)	La carga debe hacerse de manera segura	No es posible acceder a los datos personales de los usuarios salvo el administrador del sistema, que tampoco puede acceder a las contraseñas.	**AT008**
9	Agentes	Accede a la aplicación	Tiempo de ejecución	Agents	Cada agente puede acceder a sus datos, pero no a los datos de otros agentes	El acceso a los datos se permite solamente cuando la información de email/contraseña son correctas.	**AT009**

Escenario Nº	Fuente de estímulo	Estímulo	Entorno	Artefacto	Respuesta	Medición de la respuesta	Atributo de calidad afectado
10	Administrador del Sistema	Carga un fichero Excel en la base de datos	Tiempo de ejecución	Parser, DBUpdate y ReportWriter	El proceso de carga se realiza de una forma fiable y es posible chequear que los datos han sido cargados adecuadamente.	No hay errores en la base de datos ni registros duplicados. Ningún agente tiene menos información que la requerida.	**AT010**
11	Administrador del Sistema	Ejecución del sistema	Tiempo de ejecución	Agents, Loader, InciDashboard, InciManager	El proceso de carga de usuarios y de ejecución de la aplicación se comporta de una forma habitual y las opciones son fáciles de comprender	El sistema muestra ayuda si el usuario la solicita. Los mensajes de error y otra información son comprensibles por personal técnico	**AT011**
12	Sistema de Agentes	Accede al servicio Web	Tiempo de ejecución	Agents	El sistema de Agentes solicita información sobre un usuario pasando una combinación de email y contraseña	Se envía respuesta 200 OK si la combinación aparece en el sistema o error en caso contrario	**AT012**
13	Desarrollador	Implementa el sistema	Desarrollo	Agents, Loader, InciManager, InciDashboard	Los desarrolladores pueden implementar el sistema	El sistema puede implementarse en 2 semanas	**AT013**
14	Administrador del Sistema	Despliega el sistema	Despliegue	Agents, InciManager, InciDashboard	El sistema es desplegado en un entorno de producción	El sistema puede desplegarse en menos de una hora	**AT014**
15	Sistema de Agentes	Enviar correos a usuarios	Tiempo de ejecución	Loader	El sistema de Loader enviará un correo personalizado en texto plano a un usuario previamente seleccionado	Se recibirá un correo con la información escrita que se desee enviar	**TC005**
16	Sistema de incidencias	Enviar una incidencia por un agente no registrado	Tiempo de ejecución	InciManager, Agents	El gestor de incidencias recibirá la petición de enviar una incidencia. Solicitará acceso al módulo Agents que nos dirá si el agente está o no registrado en el sistema.	Si el agente no está en el sistema no se le permitirá enviar la incidencia.	**AT015**
17	Sistema de incidencias	Enviar/procesar una incidencia	Tiempo de ejecución	InciManager, InciDashboard	El tráfico y procesamiento de incidencias debe realizarse en un tiempo razonable	La información es recibida en menos de 15 seg. A cualquier hora del día.	**AT016**

Escenario Nº	Fuente de estímulo	Estímulo	Entorno	Artefacto	Respuesta	Medición de la respuesta	Atributo de calidad afectado
18	Sistema de incidencias	Enviar una incidencia en cualquier momento.	Tiempo de ejecución.	InciManager, InciDashboard	El Sistema sera capaz de asumir la rececpción de una incidencia en cualquier momento.	El Sistema kafka estará disponible 24/7	**AT016**
19	BORRADO						**AT018**
20	Sistema Dashboard	Visualizar gráficamente las incidencias	Tiempo de ejecución	InciDashBoard	El sistema le permitirá al operario ver y gestionar las incidencias desde una interfaz gráfica.	El operario puede acceder en cualquier momento a la interfaz para gestionar las incidencias.	**AT019**
21	Sistema Dashboard	Disponibilidad de la visualización de incidencias	Tiempo de ejecución	InciDashBoard	El sistema permitirá que lia visualización de incidencias esté disponible 364/365.	El sistema permite el acceso a las incidencias visualizarlas este disponible 364/365.	**AT019**
22	Sistema de incidencias	Permitir dar una fecha de llegada a las incidencias	Tiempo de ejecución	InciManager, InciDashboard	El sistema permitirá dar una fecha de llegada de la incidencia	Los agentes podrán enviar las incidencias, con una fecha de llegada asociada a estas	**AT021**
23	Sistema de incidencias	Permite al usuario acceder a su histórico de incidencias	Tiempo de ejecución	InciManager	El sistema permitirá a un usuario logueado en la aplicación acceder a un histórico de sus incidencias	El agente después de loguearse correctamente en la aplicación puede marcar la opción de ver sus incidencias	**AT022**
24	Sistema de incidencias y Loader	Las incidencias y los agentes se almacenarán en una base de datos relacional	Tiempo de ejecución	Loader, InciDashboard, Agents	El sistema almacenara todos sus datos de producción en una base de datos relacional y almacenara datos de pruebas en una base de datos independiente de cada modulo	Al lanzar la aplicación Loader los agentes se almacenan en la base de datos y cada vez que llega una incidencia esta se almacena en la misma base de datos. Al ejecutar pruebas cada módulo se encarga de almacenar sus datos de prueba en su base de datos para pruebas	**AT023, TC002, OC002**
25	Sistema Dashboard	Las incidencias estarán disponibles para que los operarios puedan gestionarlas	Tiempo de ejecución	InciDashBoard	El dashboard permitirá al operario gestionar sus incidencias asignadas 364/365	Un operario logueado en la aplicación puede gestionar sus incidencias (cambiar el estado, caducarlas …)	**AT024**

Escenario Nº	Fuente de estímulo	Estímulo	Entorno	Artefacto	Respuesta	Medición de la respuesta	Atributo de calidad afectado
26	Sistema de incidencias	La curva de aprendizaje debe ser leve	Tiempo de ejecución	InciDashboard	El sistema será fácil de aprender tanto por usuarios como por operarios	El sistema consta de módulos fácilmente manejables.	**AT025**
27	Sistema de incidencias	El servicio web es un servicio rest y el formato de entradas es json	Tiempo de desarrollo	InciManager, Agents	El sistema está diseñado con un servicio web rest que recibe peticiones post de los agentes en formato json	Se ha implementado correctamente	**TC003**
28	BORRADO						
29	Sistema de carga de datos	La carga de datos de los agentes se realizará a través de un fichero Excel	Tiempo de ejecución	Loader	El sistema carga los datos de los agentes a partir de un Excel	El formato es correcto	**TC004**
30	Sistema de pruebas	Se realizarán pruebas, ya sean unitarias, pruebas con selenium, de carga, así como historias de usuario	Tiempo de desarrollo	Loader, InciManager, Agents, InciDashboard	El sistema está provisto de pruebas, así como de integración continúa debido al uso de travis a través de la plataforma Github que será la utilizada para el control de versiones	Las pruebas pasarán correctamente	**OC003, TC006**
31	Sistema de incidencias	La tecnología utilizada para el envío de incidencias entre los módulos será ApacheKafka mediante su API cloud (CLOUD KARAFKA)	Tiempo de desarrollo	InciManager, InciDashBoard	El sistema usará dicha tecnología para el envío de incidencias entre los módulos para soportar un gran volumen de envío de datos	La tecnología soporta correctamente el envío de incidencias	**TC008**
32	Escenario Obvio	------	-------	-------	-------	-------	**TC001**
33	Escenario Obvio	------	-------	-------	-------	-------	**TC007**

Tabla 6. Lista de escenarios de calidad

9 Vistas

En los próximos párrafos se describirán algunas de las vistas identificadas y se documentarán de acuerdo con las instrucciones definidas en la guía de aprendizaje.

Vista	Stakeholders	Restricciones	Atributos de calidad	Escenarios
Context	ST-01, ST-02, ST-03, ST-04, ST-05, ST-06	TC001, TC002, TC006	AT008, AT011, AT013	8, 11, 13, 24, 30 , 32
Loader	ST-01, ST-02 ST-03, ST-04, ST-06	TC001, TC002, TC004, TC005, TC006, OC002	AT003, AT004, AT007, AT008, AT010, AT011, AT013, AT023	3, 4, 7, 8, 10, 11, 13, 15, 24, 29, 30 , 32
Agents	ST-01, ST-02, ST-03, ST-04, ST-06	TC001, TC002, TC003, TC006, TC007	AT001,AT002, AT006, AT008,AT009, AT011, AT012,AT013, AT014, AT015	1,2,6,8,9,11, 12,13,14,16, 24, 27, 30, 32, 33
InciManager	ST-01, ST-02, ST-04, ST-06	TC001, TC002, TC006, TC007, TC008	AT001,AT008, AT011, AT013, AT014,AT015, AT016, AT021, AT022	1,8,11,13,14,16,17, 18, 22,23,24,30,31, 32, 33
InciDashboard	ST-01, ST-02, ST-05, ST-06	TC001, TC002, TC006, TC007, TC008	AT001,AT008, AT011, AT013, AT014,AT016, AT019, AT021,AT023, AT024, AT025	1, 8, 11, 13, 14, 17,18, 20, 21, 22, 24, 25, 26, 30, 31, 32, 33
Paquetes	ST-01, ST-02, ST-06	TC002, TC008, OC003	AT023	24, 31
Despliegue	ST-01, ST-02, ST-06	TC008	AT001, AT014, AT016	1, 14, 18, 31

9.1 Contexto

La vista de sistema describe los cuatro subsistemas en interacción, así como sus interfaces.

9.1.1 Presentación principal

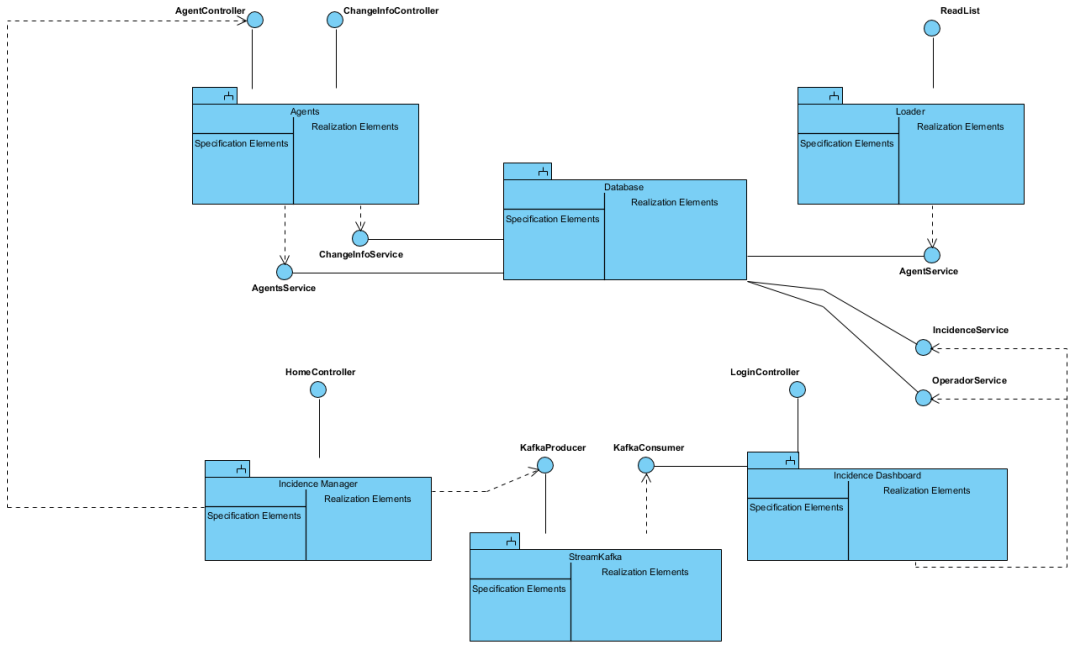


Figura 2. Context view

9.1.2 Catálogo de elementos

9.1.2.1 *Elementos*

Elemento	Propiedades
Loader	Se encarga de la introducción de las listas de agentes, así como sus tipos, en el sistema. Lee un fichero xls con los datos de los agentes y un fichero csv con los tipos de estos. Crea las claves. Añade los emails para los usuarios dados de alta.
Agents	Es el módulo usado por los agentes para comprobar que han sido dados de alta y opcionalmente para hacer el cambio de clave u otros datos. También lo utilizarán para enviar las incidencias al sistema.
DataBase	Este módulo encapsula los accesos a la base de datos.
IncidenceManager	Módulo encargado de enviar (producir) incidencias a través de Kafka. Se comunica con Agents para recoger agentes de la base de datos a través de peticiones POST en formato JSON.
StreamKafka	Módulo que conecta, en tiempo real, el módulo IncidenceDashboard e IncidenceManager

9.1.2.2 *Relaciones*

Los datos de los agentes se introducen en el sistema a través de la interface *ReadList* del módulo *Loader*. Para cada usuario, se crea una clave y se emite un email con todos los datos del usuario. Los tipos de agentes se cargan desde un fichero en formato CSV.

Posteriormente se envían a la base de datos a través del servicio *AgentService*.

El módulo *Agents* permite al usuario entrar en sesión pidiendo los datos al módulo *DataBase* a través de la interfaz *AgentsService*. *Agents* también permite cambiar datos de agentes enviando peticiones web a través del controlador ChangeInfoController.

InciManager, por su parte, envía peticiones POST a métodos del controlador *AgentController* del módulo *Agents*. Por ejemplo, el método *showInfo* del controlador devolverá un agente que encontrará en la base de datos. Permitirá, por tanto, loguearse a agentes a través de peticiones web en el controlador *HomeController*, permitiéndoles enviar incidencias a través de la interfaz *KafkaProducer*, y ver un histórico de todas las incidencias que ha enviado.

El módulo *InciDashboard* muestra la lista de incidencias que tiene cada operario asignadas, y les permite gestionarlas. El módulo *InciManager* será el encargado de enviar las incidencias a través del *producer* de *Kafka*, y *Dashboard*, a través de la interfaz Kafka*Consumer*, las guardará en la base de datos y las asignará a los operarios.

Se podrán ver estadísticas, de forma gráfica, de las incidencias almacenadas en el sistema.

9.1.2.3 *Interfaces / Puertos*

9.1.2.3.1 Loader

Interface	Tipo	Tecnología	Propiedades
ReadList	Interface	Invocación mediante línea de comandos	Se invocará como un programa en consola

9.1.2.3.2 Agents

Interface	Tipo	Tecnología	Propiedades
AgentController	Interface	Servicio Web	Controlador que implementa métodos para peticiones HTTP relacionadas con mostrar información de agentes.
ChangeInfoController	Interface	Servicio Web	Controlador que implementa métodos para peticiones HTTP relacionadas con el cambio de datos en agentes.

9.1.2.3.3 DataBase

Interface	Tipo	Tecnología	Propiedades
AgentService	Interface	Invocación a Método	Inserta agentes en la base de datos.
AgentsService	Interface	Invocación a Método	Devuelve datos de agentes en la base de datos.
ChangeInfoService	Interface	Invocación a Métodos	Actualiza datos de agentes en la base de datos.
OperadorService	Interface	Invocación a Método	Devuelve operarios de la base de datos.
IncidenceService	Interface	Invocación a Método	Devuelve o inserta incidencias en la base de datos.

9.1.2.3.4 IncidenceManager

Interface	Tipo	Tecnología	Propiedades
HomeController	Interface	Invocación a Método	Permite, a través de peticiones HTTP, entrar en la vista principal del agente logueado.

9.1.2.3.5 StreamKafka

Interface	Tipo	Tecnología	Propiedades
KafkaProducer	Interface	Invocación a Método	Produce mensajes al *stream* de Kafka
KafkaConsumer	Interface	Invocación a Método	Consume mensajes al *stream* de Kafka.

9.1.2.4 *Comportamiento*

9.1.2.4.1 Loader

Permite cargar agentes desde un archivo Excel, y guardarlos en la base de datos.
Se ejecuta de forma totalmente independiente del resto de módulos.

9.1.2.4.2 Agents

Permite a los usuarios poder acceder al sistema para comprobar que han sido dados de alta, usando la información recibida en el email. Los usuarios podrían no acceder directamente mediante un navegador Web, sino a través de un sistema externo que invoca el módulo como un servicio Web.

9.1.2.4.3 DataBase

Este módulo encapsulará las operaciones de acceso a la base de datos así como la tecnología a utilizar.

9.1.2.4.4 IncidenceManager

Permite enviar incidencias en tiempo real a agentes registrados en la aplicación. A los agentes que las envían se les pedirá cierta información de las incidencias, como la localización, el título, la descripción, una lista de campos clave-valor o etiquetas.

9.1.2.4.5 IncidenceDashboard

Permite asignar incidencias a operarios para su posterior gestión por estos. Permite, además, gestionar qué campos de incidencias son críticos (solo los administradores). Las incidencias enviadas en tiempo real actualizan la lista de incidencias de los operarios. Además, se permiten ver estadísticas de las incidencias del sistema.

9.1.2.4.6 StreamKafka

Módulo que permite enviar incidencias en tiempo real, entre el módulo IncidenceManager (producer) e IncidenceDashboard(consumer).

9.1.2.5 Justificación de las decisiones

Las decisiones que han llevado a este diseño son:

Escenario	Atributos de calidad	Justificación
8	AT008	Las contraseñas de todos los usuarios de la aplicación están debidamente encriptadas con funciones hash de Java, con una sal concreta.
13	AT013	El procesamiento y envío de incidencias debe hacerse de forma rápida, permitiendo enviar varias de forma continuada y por distintos agentes.
24	TC002	Escenario obvio.
30	TC006	Cada módulo tiene sus propias pruebas. Los módulos que tienen interfaz gráfica de usuario, tienen pruebas de Selenium y de Cucumber, además de pruebas con Junit para probar repositorios y modelo. Los que no tienen, contienen una serie de pruebas unitarias que comprueban el correcto funcionamiento de los servicios y repositorios de acceso a datos, así como las clases del modelo.

9.2 Vista de Loader

La vista de *Loader* muestra el primer nivel de descripción de los componentes.

9.2.1 Presentación principal

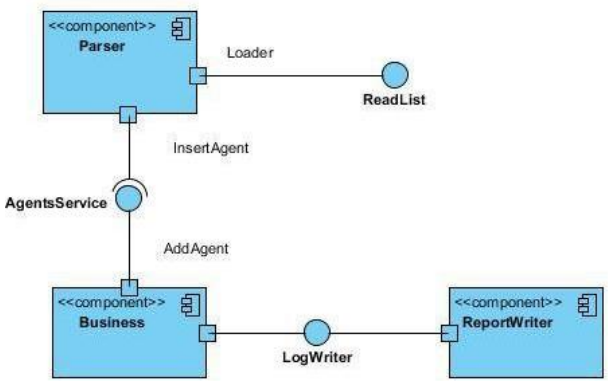


Figura 3. Vista loader

9.2.2 Cátalogo de elementos

9.2.2.1 *Elementos*

Elemento	Propiedades
Parser	Lee los datos de entrada del Excel (así como del fichero CSV que contiene los tipos de agentes), y los transforma en un contenedor de objetos que puede ser recorrido para su inserción en la base de datos. También crea el *usuario/password* de los agentes, y el identificador usado para la comunicación. Durante el diseño y la implementación hay que partir este componente en los subcomponentes necesarios para separar todos estos servicios y hacerlo de manera que se cumplan los atributos de calidad AT002, AT003, AT004 y AT007.
Business	Encapsula todas las operaciones de base de datos usando interfaces para permitir el acceso a la base de datos.
ReportWriter	Recibe cadenas de información con los datos del usuario que fue Imposible dedar de alta y las razones de dicho fallo y escribe un registro en un fichero de texto secuencial, indicando toda la información necesaria para poder revisar visualmente los fallos.

9.2.2.2 *Relaciones*

El componente *Parser*recibe los ficheros de entrada *Excel*y *CSV*y, mediante un *parser*específico para cada fichero, los convierte en objetos. Añade a estos objetos el identificador (usuario) y el *password*, y lo añade a la base de datos.

Si se producen errores en la carga de datos (DNI duplicados, campo DNI vacío, etc.) o si el componente *delabasededatos*devuelve un error, esta información se escribe en un fichero de *LOG*mediante la interface *LogWriter* y el componente Report*Writer*.

Si aparecen otras situaciones de error se pueden documentar usando el mismo componente *ReportWriter*.

9.2.2.3 *Interfaces / Puertos*

9.2.2.3.1 Parser

Interface	Tipo	Tecnología	Propiedades
ReadList	Interface	Invocación a Métodos	Lee el fichero de *Excel* con los datos de una lista de ciudadanos.
Loader	Port		Crea los subcomponentes del *parser* necesarios para procesar el fichero de entrada.
InsertAgent	Interface	Invocación a Métodos	Accede al servicio de AgentService para añadir el agente a la base de datos

9.2.2.3.2 Business

Interface	Tipo	Tecnología	Propiedades
AgentService	Interface	Invocación a Métodos	Recibe un objeto con la información para insertar/modificar/eliminar en la base de datos.
AgentServiceImpl	Port		En cada método llama a los métodos execute de la interfaz Command.
Command	Interface	Invocación a Métodos	Interfaz encargada de definir el método execute, que redefinirán las clases definidas en AgentService.
CommandExecutor	Port		Clase que se encarga de ejecutar el método execute() de una clase de tipo Command, de tal manera que las operaciones llevadas a cabo por el execute() queden persistentes en la base de datos, abriendo un contexto de persistencia.

9.2.2.3.3 ReportWriter

Interface	Tipo	Tecnología	Propiedades
LogWriter	Port		Clase estática que añade al fichero log una línea pasada por parámetro.

9.2.2.3.4 Parser

Introduce las listas de ciudadanos en el sistema a partir de ficheros Excel formados por filas de ciudadanos, cada una con la siguiente información (excepto la primera fila que contiene las

cabeceras):

- Nombre (y apellidos para las Personas) : String
- Email (con un formato acorde a las convenciones de correo electrónico) : String
- Tipo (representa el tipo de agente) : int
- Identificador (será único para cada agente, así como su nombre de usuario) : String
- Localización (opcional para las Personas y Entidades) : String

La invocación se hará mediante un programa *batch* ejecutado en línea de comando por el administrador del sistema. Durante la importación las listas de ciudadanos, se creará un usuario por cada ciudadano, cuyo nombre de usuario coincidirá con el correo electrónico y se generará una contraseña aleatoria. La combinación adecuada de email/contraseña permitirá al usuario entrar al sistema, acceder a su información y participar en el portal.

Este componente también creará los emails personales comunicando al usuario que ha sido añadido al Sistema de Incidencias, e informando de su clave de acceso.

9.2.2.3.5 DBUpdate

Actualiza la base de datos. Ver 9.1.2.4.3.

9.2.2.3.6 ReportWriter

Guarda en un fichero de texto la información de los errores producidos en el proceso de conversión. La información básica a guardar es:

- Identificador del usuario del que se ha obtenido el error
- Descripción del error (con toda la información necesaria)

9.2.3 Diagrama contextual

Ver 9.1.

9.2.4 Justificación de las decisiones

Las decisiones que han llevado a este diseño son:

Escenario	Atributos de calidad	Justificación
3	AT003	Prever una interfaz y un objeto que pueda estar vacío para el informe de errores (*WriteReport*) facilita la modificabilidad en caso de añadir nuevos tipos de registros posteriormente.
4	AT004	Prever el posible cambio a otro formato de salida.
7	AT007, TC004	Está diseñado para cargar los datos un fichero Excel y un fichero CSV.El Parser recibe estos ficheros y los parsea.
8	AT008	La utilización de una base de datos relacional con acceso mediante SQL puede permitir a los alumnos verificar que los datos han sido cargados adecuadamente
10	AT010	El sistema permite al administrador chequear que los usuarios se han cargado correctamente mediante la ejecución de las pruebas.

11	AT011	La utilización de una aplicación batch que pueda ser ejecutada manualmente o configurada para su ejecución automatizada es una práctica común entre los administradores de sistemas.
13	AT013	El sistema ha sido diseñado de tal forma que su implementación ha sido bastante sencilla
15	TC005	El sistema provee de un mecanismo para simular un envio de correo a un usuario cuando sus datos son validados
24	AT023, TC002, OC002	El sistema almacena todos los datos de los usuarios en una base de relacional.
30	OC003, TC006	El sistema esta provisto de pruebas para asegurar que el sistema funciona correctamente

9.3 Vista de Agents

9.3.1 Presentación principal

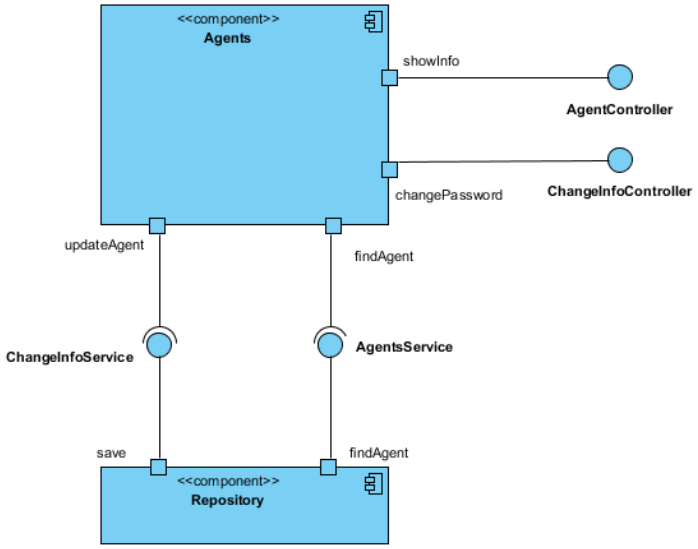


Figura 4. Vista de Participants

9.3.2 Catálogo de elementos

9.3.2.1 *Elementos*

Elemento	Propiedades
Agents	Se accede a través de una petición POST (usuario, contraseña, kind) y devuelve una cadena de texto en formato JSON cuyo contenido será diferente en caso de que exista o no. Además, también implementa la posibilidad de cambiar la clave del agente y ver su información mediante una interfaz de usuario.
Repository	Componente que accede a la base de datos.

9.3.2.2 *Relaciones*

El Sistema de Incidencias invoca *Agents* enviando una petición POST (usuario, contraseña y kind) y se comprueba si dicho usuario existe. Para ello, se invoca al servicio *AgentsService* que, a su vez, llama al repositorio, que busca el agente en la base de datos. Se devuelve un JSON con los datos del agente o un código de error en caso de no haya tenido éxito la petición.

Además, se incluye la opción de que un agente pueda cambiar su contraseña (clave) si así lo desea mediante el servicio ChangeInfoService, que al igual que el AgentsService, accede a la base de datos a través de la interfaz Repository.

Se pueden crear tantas interfaces como elementos a modificar o usar la anterior con algún tipo de código para definir los datos a modificar.

9.3.2.3 *Interfaces / Puertos*

9.3.2.3.1 Agents

Interface	Tipo	Tecnología	Propiedades
AgentController	Interface	Invocación a Métodos	Recibe una petición POST y genera una cadena de texto en formato JSON con los datos del agente en caso de éxito o un código de error en caso contrario.
AgentsService	Interface	Servicio Web	Implementa la lógica de negocio a la hora de buscar un agente.
ChangeInfoController	Interface	Invocación a Métodos.	Permite el cambio de contraseña (clave) de un agente.
ChangeInfoService	Interface	Servicio Web	Implementa la lógica de negocio para actualizar la contraseña de un agente.
showInfo	Port		Recoge los datos de la petición POST y se los envía al servicio *AgentsService*. Además lanza código de error en caso de que los parámetros no sean válidos o no exista el agente.
findAgent	Port		Implementado en el servicio *AgentsService,* llama al repositorio que accede a la base de datos para buscar el agente especificado.
updateAgent	Port		Implementado en el servicio *ChangeInfoService*, actualiza la contraseña de un agente haciendo una llamada al método correspondiente en AgentRepository.
changePassword	Port		Recoge los datos de la petición para el cambio de contraseña y se la manda al servicio *ChangeInfoService*.

9.3.2.3.2 Repository

Interface	Tipo	Tecnología	Propiedades
AgentRepository	Interface	Repositorio	Se encarga de comunicar el módulo agents con la base de datos.
findAgent	Port		Implementado en el repositorio *AgentRepository* busca un agente en la base de datos a partir de su nombre de usuario, contraseña y kind. Si tiene éxito devuelve el agente especificado
save	Port		Actualiza la información del agente (contraseña)

9.3.2.4 *Comportamiento*

9.3.2.4.1 Agents

Ver 9.3.2.2.

Implementa un servicio web REST para gestionar las peticiones de información sobre los usuarios. La petición principal será una petición HTTP POST que se realizará a la dirección:

http://35.180.34.205:8070/info

La petición POST contiene datos JSON con la siguiente estructura:
{"login": email, "password": password, "kind": tipo usuario}

En caso de que la combinación login/password/kind aparezca en la base de datos, la respuesta será 200 OK con el cuerpo JSON de la forma:

```
{
        "id": "id_usuario" (long),
        "password": "password",
        "kind": "tipo_usuario",
        "kindCode": "id_kind" (long),
        "dni": "dni",
        "nombre": "nombre",
        "apellidos": "apellidos",
        "email": "email",
        "username": "nombre_usuario"
}
```

En caso de que la combinación (email, password, kind) no aparezca, la respuesta será "404 Not Found". Si los parámetros no son correctos se devuelve un código de error HTTP 406.

Se ha implementado un interfaz HTML para que el servicio Web pueda también ser utilizado por personas a través de un navegador Web convencional.

El servicio Web ha sido extendido para permitir a los usuarios cambiar su password.

9.3.2.4.2 Repository

Encapsula todos los accesos a la base de datos, buscar agente por usuario, contraseña y kind y actualizar la clave.

9.3.3 Diagrama contextual

Ver 9.1.

9.3.4 Justificación de las decisiones

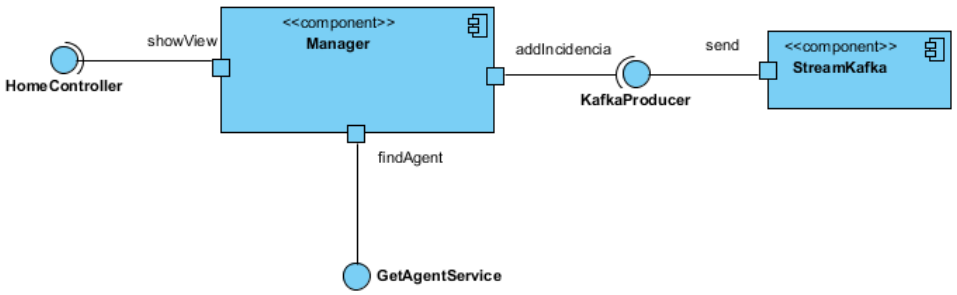
Las decisiones que han llevado a este diseño son:

Escenario	Atributos de calidad	Justificación
1	AT001	La utilización de un servicio web REST aprovecha de la tecnología HTTP y facilita el despliegue del sistema en infraestructuras de alta disponibilidad como pueden ser servidores Web, tanto locales como en la nube.
2	AT002	Debido al buen diseño del Parser, realizar un cambio en dicho elemento resultaría una tarea bastante sencilla.
6	AT006	La utilización del framework Spring Boot facilitará el desarrollo posterior de características comunes de la web como la negociación de contenido, dado que el framework ya contiene herramientas para su implementación.
8	AT008	La restricción de acceso mediante *email/password/kind* se considera suficientemente segura para este proceso. Las claves deberán almacenarse encriptadas.

9	AT009	El desarrollo de un servicio web REST basado en formatos JSON facilitará la creación de pruebas. El framework Spring Boot contiene varias herramientas para pruebas unitarias y de integración.
11	AT011	La utilización de una aplicación batch que pueda ser ejecutada manualmente o configurada para su ejecución automatizada es una práctica común entre los administradores de sistemas.
12	AT012	El uso de un servicio web REST permitirá el acceso automático al sistema a través de software cliente.
13	AT013	El sistema ha sido diseñado de tal forma que su implementación ha sido bastante sencilla
14	AT014	La utilización del framework Spring Boot facilita el despliegue. Hay varios ejemplos que muestran cómo desplegar aplicaciones basadas en Spring Boot en servidores de producción.
16	AT015	Si la combinación usuario, contraseña y kind no es correcta, se redirige a una página de error al iniciar sesión.
24	TC002	El sistema usa una base de datos relacional para almacenar los datos. Ya que el equipo de desarrollo está más familiarizado con este tipo de base de datos.
27	TC003	El Formato de entrada debe ser un fichero JSON, ya que se trata de un formato muy sencillo , fácil de entender y de lo más utilizado hoy en día
30	TC006	El sistema tiene una batería de pruebas, entre las que se incluye Selenium, Cucumber, Junit y pruebas de carga con Gatling. Entre todas se consiguen probar las entidades del modelo de la aplicación, los controladores, los servicios de acceso a datos, el parseador de incidencias y la estabilidad del sistema ante el acceso de un cierto número de usuarios simultáneos.
33	TC007	Spring-boot proporciona una arquitectura impuesta Modelo Vista Controlador, que permite definir una estructura bien definida en el sistema.

9.4 Vista de Incidence Manager

9.4.1 Presentación principal



9.4.2 Catálogo de elementos

9.4.2.1 *Elementos*

Elemento	Propiedades
Manager	Envía una petición al módulo Agents para comprobar que el agente está registrado, y si existe, permite crear nuevas incidencias y consultar el historial de las enviadas por este.
StreamKafka	Se envía una cadena de texto al módulo InciDashboard con todos los datos de una incidencia.

9.4.2.2 *Relaciones*

El componente Manager, se comunica con Agents para chequear que un agente esté registrado en el sistema. Para ello se envía una petición POST al módulo Agents a través de GetAgentService. Una vez que el agente es chequeado de forma correcta, éste puede enviar una incidencia a través de Apache Kafka mediante la interfaz KafkaProducer para que el módulo InciDashboard la gestione. Por último un agente puede hacer un seguimiento de las incidencias que ha enviado.

9.4.2.3 *Interfaces / Puertos*

9.4.2.3.1 Manager

Interface	Tipo	Tecnología	Propiedades
GetAgentService	Interface	Servicio Web	Envía una petición al módulo Agents para comprobar que el agente está registrado en el sistema.
HomeController	Interface	Controlador	Punto de entrada al módulo InciManager.
showView	Port		Muestra la vista para que un agente inicie sesión.
findAgent	Port		Implementado en GetAgentService, se encarga de hacer la petición al módulo Agents para validar que el agente está registrado en el sistema.
addIncidencia	Port		Un agente crea una incidencia que será enviada por ApacheKafka

9.4.2.3.3 StreamKafka

Interface	Tipo	Tecnología	Propiedades
KafkaProducer	Interface	Invocación a Métodos	Se encarga de enviar la incidencia al módulo inciDashboard.
send	Port		Implementado en KafkaProducer, se encarga de enviar la incidencia por ApacheKafka al módulo inciDashboard.

9.4.2.4 *Comportamiento*

9.4.2.4.1 Manager

Realiza una petición POST con los datos de un agente, al módulo Agents, para comprobar que efectivamente, el agente se encuentra registrado en el sistema. Una vez la petición ha tenido éxito, el agente inicia sesión en el módulo.
Cuando el agente ya se encuentra en el sistema, podrá crear una nueva incidencia indicando su nombre, descripción, una lista de etiquetas y campos y la localización (manual o automáticamente). Además, también podrá consultar el listado de incidencias que ha realizado hasta ese momento.

9.4.2.4.2 StreamKafka

StreamKafka se encarga de enviar la cadena de texto que contiene los datos de la incidencia, del componente Manager al módulo InciDashboard, para su posterior gestión.

9.4.3 Diagrama contextual

Ver 9.1.

9.4.4 Justificación de las decisiones

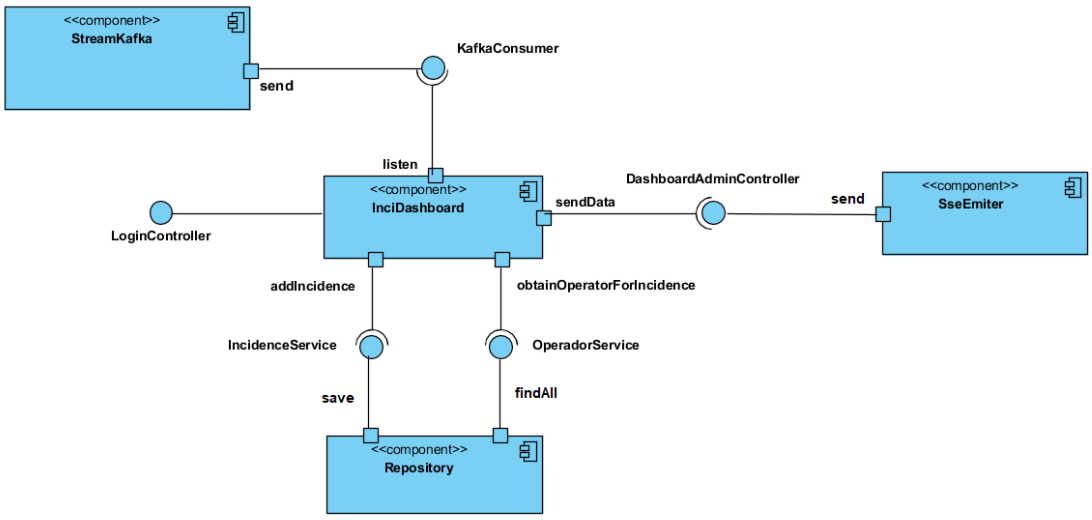
Las decisiones que han llevado a este diseño son:

Escenario	Atributos de calidad	Justificación
1	AT001	La utilización de un servicio web aprovecha de la tecnología HTTP y facilita el despliegue del módulo en servidores Web que garantizan alta disponibilidad como en la nube.
8	AT008	La restricción de acceso mediante *email/password/kind* se considera lo suficientemente segura para este proceso. Las claves deberán almacenarse encriptadas
11	AT011	La utilización de una aplicación batch que pueda ser ejecutada manualmente o configurada para su ejecución automatizada es una práctica común entre los administradores de sistemas.
13	AT013	El API del servicio web es simple y contiene la funcionalidad mínima necesaria. La utilización del framework Spring Boot facilitará el desarrollo por los estudiantes dado que el framework tiene soluciones para toda la funcionalidad requerida.
14	AT014	La utilización del framework Spring Boot facilita el despliegue. Hay varios ejemplos que muestran cómo desplegar aplicaciones basadas en Spring Boot en servidores de producción.
16	AT015	Un agente que no se encuentre registrado en el sistema no podrá enviar incidencias al mismo. Se consigue, por tanto, que solo usuarios autorizados puedan interaccionar con el sistema, denegando el acceso a otro usuarios con malas intenciones (robo de datos, envío de incidencias fraudulentas, etc.).
17	AT016	El procesamiento y envío de incidencias debe hacerse de forma rápida, permitiendo enviar varias de forma continuada y por distintos agentes.
22	AT021	Ciertas incidencias deben poder tener una fecha de caducidad. Un ejemplo claro sería cuando un sensor envía datos con la temperatura en una hora específica. Se podría establecer que esos datos que envía el sensor tienen una hora de duración antes de caducar.

Escenario	Atributos de calidad	Justificación
23	AT022	Una vez que el agente se encuentre logueado en la aplicación, tendrá una opción en la vista para ver su historial de incidencias.
24	TC002	El sistema usa una base de datos relacional para almacenar los datos. Ya que el equipo de desarrollo está más familiarizado con este tipo de base de datos
27	TC006	El sistema tiene una batería de pruebas, entre las que se incluye Selenium, Cucumber, Junit y pruebas de carga con Gatling. Entre todas se consiguen probar las entidades del modelo de la aplicación, los controladores, los servicios de acceso a datos, el parseador de incidencias y la estabilidad del sistema ante el acceso de un cierto número de usuarios simultáneos.
33	TC007	Spring-boot proporciona una arquitectura impuesta Modelo Vista Controlador, que permite definir una estructura bien definida en el sistema
31	TC008	El sistema envía incidencias a través de Kafka. Para su despliegue online, se ha utilizado Cloud Karafka, que permite crear *Topics* en la nube y, de una forma sencilla, configurarlo en nuestra aplicación. Dicha configuración está contenida en la clase KafkaConsumerFactory

9.5 Vista de Incidence Dashboard

9.5.1 Presentación principal



9.5.2 Catálogo de elementos

9.5.2.1 *Elementos*

Elemento	Propiedades
InciDashboard	Componente que se comunica con Kafka para recibir incidencias, con la base de datos para almacenarlas y para recoger operarios de ella (para asignarles incidencias), y con el componente SseEmitter, encargado de enviar datos a los Emitter suscritos al *Topic* al que se ha enviado la incidencia.
StreamKafka	Componente encargado de enviar las incidencias al consumer de Kafka, para su posterior procesamiento por InciDashboard.
Repository	Componente que accede a datos de la base de datos.
SseEmitter	Componente encargado de "suscribirse" a *Topics* de Kafka, de tal forma que, cuando se envié nueva información a dichos *Topics*, se lanzará un evento automáticamente en la aplicación.

9.5.2.2 *Relaciones*

El componente *StreamKafka* es el encargado de enviar incidencias al consumer de Kafka (*KafkaConsumer*), para su posterior procesamiento en InciDashboard.

Cuando la incidencia ha sido enviada, el componente InciDashboard recibe la información correspondiente a dicha incidencia, y se encarga de asignarle un operario a través del elemento Repository. El operario asignado será aquel con menos incidencias asignadas y abiertas hasta el momento.

También se envían ciertos datos de la incidencia a los *Emitter* suscritos al *Topic* al que se envió la información desde *StreamKafka*. A través de la interfaz DashboardAdminController, envía los datos anteriormente mencionados a la lista de *Emitter*, de tal forma que permitan realizar operaciones en tiempo real sobre la aplicación tales como, por ejemplo, actualizar la vista de incidencias sin tener que recargar la página.

El sistema ofrecerá un módulo de administración que permitirá al personal administrativo configurar

el comportamiento del sistema de forma interactiva. Por ejemplo, podrá utilizarse para establecer qué tipos de incidencias se asignan a qué grupos de operarios, así como configurar los permisos que los diferentes operarios tienen para asignar o gestionar incidencias.

El sistema podrá ofrecer diferentes visualizaciones gráficas o estadísticas de las incidencias.

9.5.2.3 *Interfaces / Puertos*

9.5.2.3.1 StreamKafka

Interface	Tipo	Tecnología	Propiedades
send	Port		Envía datos a un *Topic.*
KafkaConsumer	Interface	Invocación a métodos	Recibe el envío de datos a *Topics* desde el componente *StreamKafka*. Las operaciones para el envío de los datos quedan registradas en un fichero log.

9.5.2.3.2 InciDashboard

Interface	Tipo	Tecnología	Propiedades
KafkaConsumer	Interface (Requerida)	Invocación a métodos	Solicita los datos recibidos en el consumidor de *Kafka*, a un determinado *Topic*, para consumirlos posteriormente.
listen	Port		Permite interceptar todos los datos que se envían a determinados *Topic*, de tal forma que sean parseados y convertidos a incidencias.
addIncidence	Port		Permite añadir incidencias a través de servicios.
IncidenceService	Interface (Requerida)	Invocación a métodos	Solicita añadir una nueva incidencia.
obtainOperatorForIncidence	Port		Permite obtener un operario, para asignarlo a una incidencia.
OperadorService	Interface (Requerida)	Invocación a métodos	Solicita recuperar un operario. En concreto, aquel con menos incidencias asignadas en estado “abierta”.
sendData	Port		Permite enviar ciertos datos de una incidencia (su estado, su título y su ID) a los *Emitter*.
DashboardAdminController	Interface (Requerida)	Invocación a métodos	Solicita el envío de datos a cada elemento de la lista de *Emitter*.

9.5.2.3.3 Repository

Interface	Tipo	Tecnología	Propiedades
save	Port		Almacena una nueva incidencia.
IncidenceService	Interface	Invocación a métodos	Permite añadir una incidencia.
findAll	Port		Devuelve una lista con todos los operarios del sistema.
OperadorService	Interface	Invocación a métodos	Permite devolver el operario, de la lista completa de operarios, que menos incidencias asignadas tenga con estado “abierta”.

9.5.2.3.4 SseEmitter

Interface	Tipo	Tecnología	Propiedades
send	Port		Envía datos de una incidencia a los *Emitter* suscritos a un determinado *Topic*.
DashboardAdminController	Interface	Invocación a métodos	Permite enviar los datos de una incidencia a una lista de *Emitter*.

9.5.2.4 *Comportamiento*

9.5.2.4.1 StreamKafka

Componente encargado del envío de incidencias a través de *Apache Kafka*. El *producer* envía los datos desde el módulo *IncidenceManager*, mientras que el *consumer* los recibe para su consecuente parseo y almacenamiento en la base de datos.

9.5.2.4.2 InciDashboard

Se encarga de recibir las incidencias y parsearlas de una forma concreta.
Las incidencias recibidas tienen la siguiente forma:

- Nombreagente@Nombreincidencia@Descripciondelaincidencia@Latitud$Longitud@Etiquetas@Listadecamposclavevalor@Fechaenmilisegundos

La lista de campos clave valor serán, por ejemplo, Fuego:Extremo. Si se añade más de un campo, se separarán con el símbolo $.

La fecha en milisegundos puede ser sacada de esta página web.

Un ejemplo de envío de una incidencia:

- Juan@Fuego en Oviedo@El parque San Francisco está quemándose a causa de un cigarrillo mal apagado@43.3616142$5.8506767@Fuego$Parque@Temperatura:Alta$Fuego:Extremo@1521893518784

El sistema transformará esa cadena de datos en una incidencia, la guardará en la base de datos, y la asignará a un operario. Además, encapsulará el id, el estado y el título de la incidencia en una cadena JSON, que enviará a los *Emitter* suscritos a un determinado *Topic*, los cuales se encargarán de actualizar en tiempo real algunos aspectos de la aplicación.

El sistema ofrecerá un módulo de administración que permitirá al personal administrativo configurar el comportamiento del sistema de forma interactiva. Por ejemplo, podrá utilizarse para establecer qué tipos de incidencias se asignan a qué grupos de operarios, así como configurar los permisos que los diferentes operarios tienen para asignar o gestionar incidencias.

El sistema podrá ofrecer diferentes visualizaciones gráficas o estadísticas de las incidencias

9.5.2.4.3 Repository

Ver 9.1.2.4.3

9.5.2.4.4 SseEmitter

Componente que se encarga de enviar datos a los elementos *Emitter*, encargados de lanzar eventos en tiempo real con los datos que les llegan. En este caso, la información que les llega

viaja en formato JSON, con un ID, el estado y el título de la incidencia enviada por *Kafka.* Es necesario para actualizar las vistas de forma automática.

9.5.3 Diagrama contextual

Ver 9.1

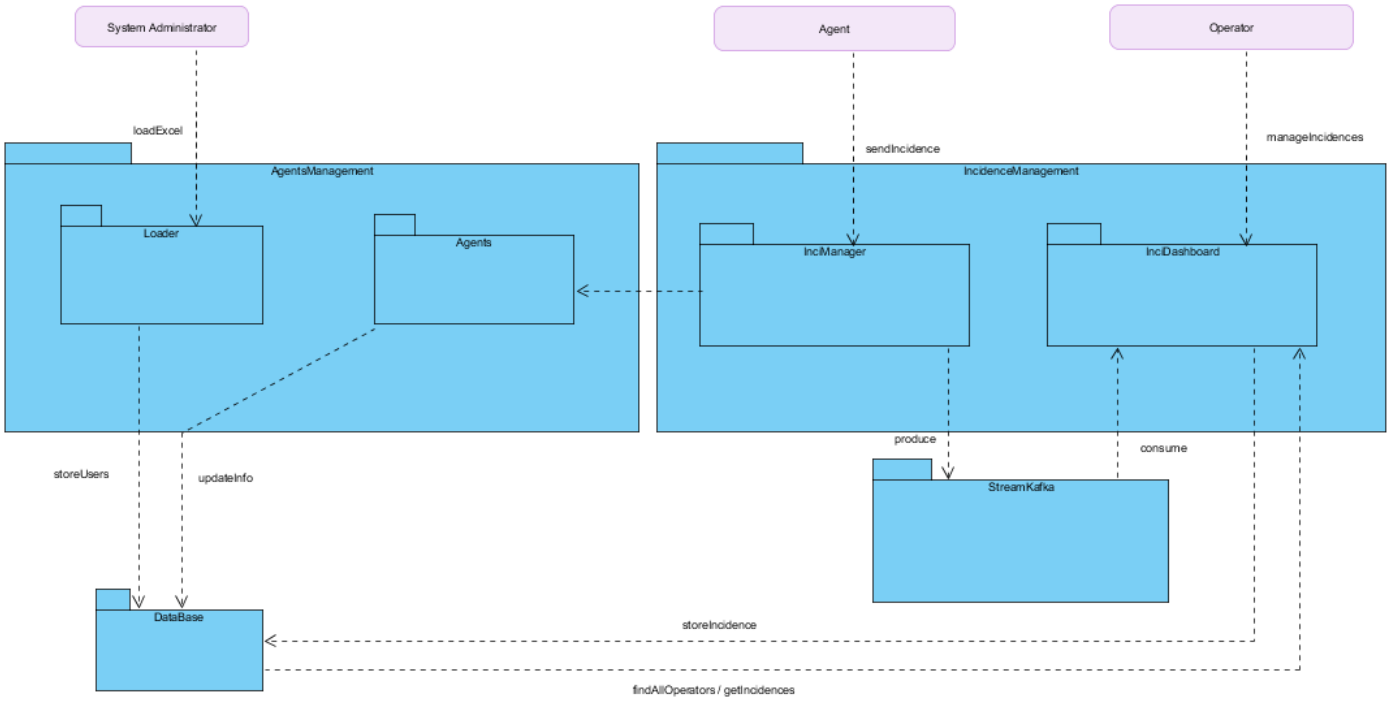
9.5.4 Justificación de las decisiones

Las decisiones que han llevado a este diseño son:

Escenario	Atributos de calidad	Justificación
1	AT001	Gracias a *AWS,* podemos disponer de una base de datos y un sistema siempre operativos. Además, Cloud Karafka también nos permite tener un sistema estable con respecto al envío de incidencias a través de *Apache Kafka.*
8	AT008	Las contraseñas de los operarios del sistema serán encriptadas gracias al módulo de *Spring-Boot Spring Security*.
11	AT011	El módulo Dashboard puede ejecutarse fácilmente desde línea de comandos sin que este módulo esté desplegado. Basta con tener Maven instalado, y ejecutar el comando *mvn spring-boot:run*.
13	AT013	El sistema es sencillo de implementar, siguiendo la arquitectura Modelo Vista Controlador (MVC), de tal forma que su modificación y expansión se haga también siguiendo dicha arquitectura.
14	AT014	El sistema se despliega de forma sencilla en AWS, con una instancia para cada módulo, haciendo que el despliegue sea ágil y rápido gracias a la separación de los módulos.
17	AT016	Las incidencias enviadas a *Topics* por *KafkaProducer* son inmediatamente almacenadas en la base de datos tras pasar por la interfaz *KafkaConsumer*. CloudKarafka hace que este proceso se haga en la nube de una forma rápida y sencilla.
18	AT016	Como se ha mencionado en el escenario anterior, CloudKarafka permite que se ejecute Kafka de forma transparente para el usuario. Además, Amazon Web Services nos permitirá desplegar los módulos y las bases de datos ágil y rápidamente.
20	AT019	El sistema permitirá ver gráficamente distintas visualizaciones de las incidencias realizadas por los usuarios, ofreciendo también distintas estadísticas de estas. Todo ello se hará en la vista de estadísticas, y cualquier operario y administrador tendrá acceso a ella.
22	AT021	El sistema provee a las incidencias de fecha de llegada para futuras gestiones
24	AT023, TC002	El sistema guardará las incidencias en una base de datos relacional MySQL, obteniendo de la misma los operarios a los que se les asignarán dichas incidencias. De las incidencias recién almacenadas se recogerán ciertos datos, que serán enviados a los *Emitter* para que actualicen determinadas vistas de la aplicación.
25	AT024	El sistema permite a los operarios gestionar las incidencias que tienen asignadas. Para ello, en la vista de listar incidencias, podrán cambiar el estado de cada una de ellas, así como ver sus detalles y escribir comentarios.
26	AT025	El sistema está preparado para ser utilizado de forma ágil e intuitiva por los operarios, con un menú superior para listar incidencias y ver las estadísticas más importantes.
30	TC006	El sistema tiene una batería de pruebas, entre las que se incluye Selenium, Cucumber, Junit y pruebas de carga con Gatling. Entre todas se consiguen probar las entidades del modelo de la aplicación, los controladores, los servicios de acceso a datos, el parseador de incidencias y la estabilidad del sistema ante el acceso de un cierto número de usuarios simultáneos.
31	TC008	El sistema recibe incidencias a través de Kafka. Para su despliegue online, se ha utilizado Cloud Karafka, que permite crear *Topics* en la nube y, de una forma sencilla, configurarlo en nuestra aplicación. Dicha configuración está contenida en la clase KafkaConsumerFactory.
33	TC007	Spring-boot proporciona una arquitectura impuesta Modelo Vista Controlador, que permite definir una estructura bien definida en el sistema.

9.6 Vista de Paquetes

9.6.1 Presentación principal



9.6.2 Catálogo de elementos

9.6.2.1 Elementos

Elemento	Propiedades
Loader	Encargado de gestionar la carga de agentes a la base de datos.
Agents	Comprueba que el agente que quiere enviar la incidencia está registrado en el sistema.
InciManager	Encargado de recibir las peticiones de envío de incidencias
InciDashboard	Se encarga de asignar la incidencia a un operario cuando es enviada por Apache Kafka.
Database	Base de datos donde se almacenan los datos
StreamKafka	Sistema que comunica los módulos InciManager y InciDashboard

9.6.2.2 Relaciones

El módulo Loader carga los agentes en DataBase a través de la interfaz storeUsers.
Cuando una petición le llega al módulo InciManager este se comunica con el módulo Agents para validar al usuario a través de la interfaz AgentsControler.
El módulo InciManager, tras validar al usuario, coge la incidencia y la envía mediante 43etic al módulo InciDashboard.
El módulo InciDashboard, cada vez que le llega una incidencia por Kafka, este módulo parseará este objeto y enviará la incidencia a la base de datos mediante la interfaz storeIncidence.

Cuando un operario se conecta al DashBoard y hace una petición para ver sus incidencias el dashboard hará una 44etición a la base de datos findAllOpertors para validar al operador y findAllIncidences para listar sus incidencias.

9.6.2.3 Interfaces / Puertos

N/A

9.6.2.4 Comportamiento

N/A

9.6.3 Diagrama contextual

Ver 9.1

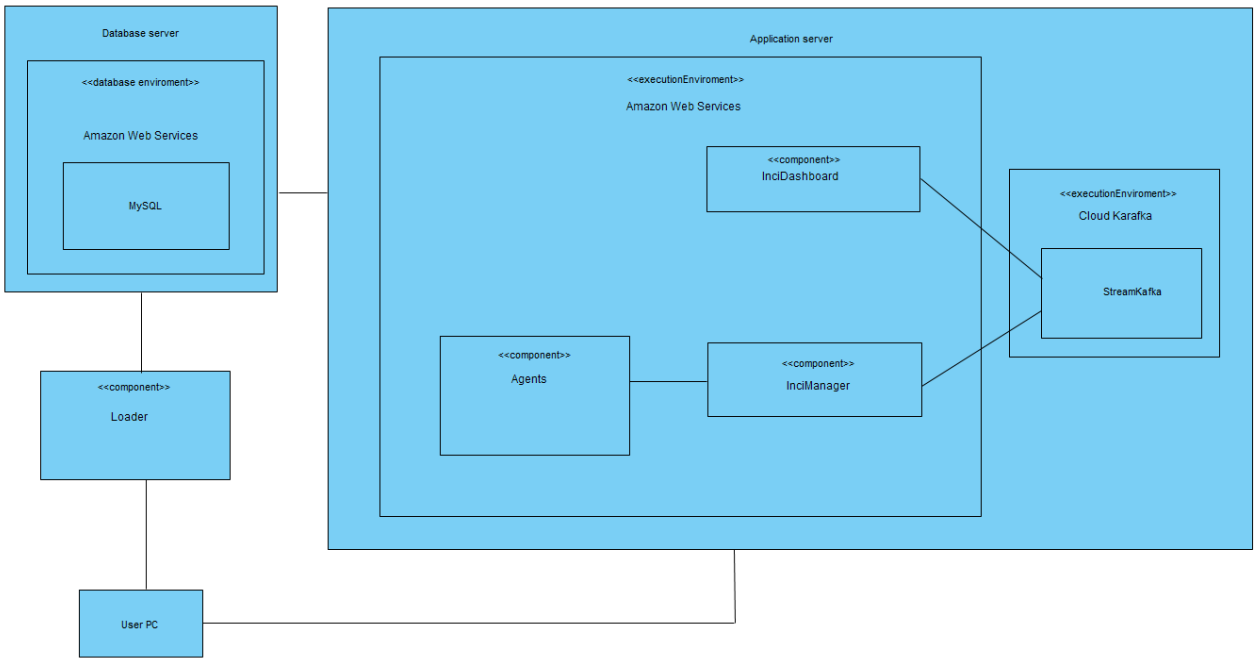
9.6.4 Justificación de las decisiones

Las decisiones que han llevado a este diseño son:

Escenario	Atributos de calidad	Justificación
24	AT023, TC002 OC003	Se emplea una base de datos relacional desplegada en AWS
31	TC008	Se emplea Kafka como entorno de ejecución para comunicar el módulo InciManager e InciDashBoard. Se emplea Cloud Karafka como entorno web de uso de Apache kafka

9.7 Vista de Despliegue

9.7.1 Presentación principal



9.7.2 Catálogo de elementos

9.7.2.1 Elementos

Elemento	Propiedades
Amazon Web Service	Entorno de ejecución donde se desplegarán los 3 módulos que necesitan despliegue.
Loader	Modulo batch que se encarga de cargar los agentes a la base datos.
DataBase Server	Base de datos relaccional encargada de almacenar todos los datos de la aplicación
Cloud Karafka	Entorno de ejecución encargado de gestionar la comunicación entre el InciManager y el InciDashboard.
Database	Base de datos donde se almacenan los datos
StreamKafka	Sistema que comunica los módulos InciManager y InciDashboard

9.7.2.2 Relaciones

El módulo Loader carga los agentes carga los agentes a la Database .
El entorno AWS se comunica con la base de datos para leer y escribir.
Los modulos InciDashBoard e InicManager se comunican mediante el entorno CloudKarafka

9.7.2.3 Interfaces / Puertos

N/A

9.7.2.4 Comportamiento

El módulo Loader carga los agentes carga los agentes a la Database comunicándose con ella a través de la interfaz StoreUsers. Los demas sitemas estan desplegados en un servidor de aplicaciones en amazon web service para su funcionamiento continuo

9.7.3 Diagrama contextual

Ver 9.1

9.7.4 Justificación de las decisiones

Las decisiones que han llevado a este diseño son:

Escenario	Atributos de calidad	Justificación
1	AT001	Haber desplegado en Amazon Web Service garantiza que realizar una petición contra el sistema ser procesada en un tiempo razonable. Además, garantiza una alta disponibilidad
14	AT014	Al escoger el sistema de despliegue elegimos Amazon Web Service dado que el tiempo de despliegue es corto y es sencillo de desplegar
18	AT016	Hemos escogido Cloud Karafka como sistema cloud de manejo de Apache kafka debido a que ofrece una alta disponibilidad y facilita el depliegue de kafka haciendolo automático.
31	TC008	Se utiliza Apache Kafka Cloud (CloudKarafka).

10 Bibliografía

ANSI/IEEE 1471. (2000). *Recommended Practice for Architectural Description of Software-Intensive Systems.* ANSI/IEEE.

Bass, L., Clements, P., & Kazman, R. (2003). *Software Architecture in Practice, Second Edition.* Boston: Addison Wesley.