

Escuela de Ingeniería Informática, Univ. Oviedo

29 de junio de 2018

Inci: Sistema de Gestión de Incidencias

**SISTEMA DE GESTIÓN DE INCIDENCIAS**

Aquilino A. Juan Fuente

José Emilio Labra Gayo

Begoña Cristina Pelayo García-Bustelo

Jordán Pascual Espada

Vicente García Díaz

*Arquitectura Software para Inci. Descripción del trabajo práctico (2018)*

Documentación de descripción de una arquitectura planteada por el alumno Alejandro Barrera Sánchez para la práctica de la asignatura Arquitectura del Software del curso 2018.

**Grado de Ingeniería Informática del Software**

|  |  |
| --- | --- |
| G:\Mis Documentos\Documentos\Universidad\Asignatures\Asig-Grado\Arquitectura del Software\Teoría\logo.png | **G:\Mis Documentos\Documentos\Universidad\Asignatures\Asig-Grado\Arquitectura del Software\Logos\logo-color.png** |

**Inci: Sistema de Gestión de incidencias**

**Autores**: **Alejandro Barrera Sánchez**

Alumno del grado Ingeniería Informática del Software en la Universidad de Oviedo. Identificador: UO251893. Con DNI 48283553K.

**Fecha**: 29 de junio de 2018

**Versión**:

Tabla de contenido

[1 Introducción y Objetivos 5](#_Toc517867488)

[2 Requisitos 6](#_Toc517867489)

[2.1 Loader 6](#_Toc517867490)

[2.2 Agents 7](#_Toc517867491)

[2.3 InciManager 7](#_Toc517867492)

[2.4 InciDashBoard 8](#_Toc517867493)

[3 Metodología usada 9](#_Toc517867494)

[4 Identificación de *Stakeholders* 10](#_Toc517867495)

[4.1 Alejandro Barrera Sánchez 10](#_Toc517867496)

[4.2 Administrador del Sistema 10](#_Toc517867497)

[4.3 Agentes 10](#_Toc517867498)

[4.4 Responsables políticos del portal 11](#_Toc517867499)

[4.5 Profesores de la asignatura 11](#_Toc517867500)

[4.6 Operarios 11](#_Toc517867501)

[4.7 Grupo de alumnos que realizaron la práctica en convocatoria ordinaria 11](#_Toc517867502)

[5 Atributos de calidad 12](#_Toc517867503)

[5.1 Lista de atributos de calidad 13](#_Toc517867504)

[5.2 Atributos de calidad e Interesados 15](#_Toc517867505)

[6 Restricciones 17](#_Toc517867506)

[6.1 Restricciones técnicas 17](#_Toc517867507)

[6.2 Restricciones organizativas 18](#_Toc517867508)

[7 Ámbito del sistema y contexto 19](#_Toc517867509)

[8 Escenarios de calidad 21](#_Toc517867510)

[9 Vistas 27](#_Toc517867511)

[9.1 Contexto 27](#_Toc517867512)

[9.1.1 Presentación principal 28](#_Toc517867513)

[9.1.2 Catálogo de elementos 28](#_Toc517867514)

[9.1.3 Diagrama contextual 31](#_Toc517867515)

[9.1.4 Justificación de las decisiones 31](#_Toc517867516)

[9.2 Loader 32](#_Toc517867517)

[9.2.1 Presentación principal 32](#_Toc517867518)

[9.2.2 Catálogo de elementos 32](#_Toc517867519)

[9.2.3 Diagrama contextual 34](#_Toc517867520)

[9.2.4 Justificación de las decisiones 34](#_Toc517867521)

[9.3 Agents 35](#_Toc517867522)

[9.3.1 Presentación principal 35](#_Toc517867523)

[9.3.2 Catálogo de elementos 35](#_Toc517867524)

[9.3.3 Diagrama contextual 37](#_Toc517867525)

[9.3.4 Justificación de las decisiones 37](#_Toc517867526)

[9.4 InciManager 38](#_Toc517867527)

[9.4.1 Presentación principal 38](#_Toc517867528)

[9.4.2 Catálogo de elementos 38](#_Toc517867529)

[9.4.3 Diagrama contextual 41](#_Toc517867530)

[9.4.4 Justificación de las decisiones 41](#_Toc517867531)

[9.5 InciDashboard 41](#_Toc517867532)

[9.5.1 Presentación principal 41](#_Toc517867533)

[9.5.2 Catálogo de elementos 42](#_Toc517867534)

[9.5.3 Diagrama contextual 43](#_Toc517867535)

[9.5.4 Justificación de las decisiones 44](#_Toc517867536)

[9.6 Vista de Paquetes 45](#_Toc517867537)

[9.6.1 Presentación principal 45](#_Toc517867538)

[9.6.2 Catálogo de elementos 45](#_Toc517867539)

[9.6.3 Diagrama contextual 48](#_Toc517867540)

[9.6.4 Justificación de las decisiones 48](#_Toc517867541)

[9.7 Vista de Despliegue 49](#_Toc517867542)

[9.7.1 Presentación principal 49](#_Toc517867543)

[9.7.2 Catálogo de elementos 49](#_Toc517867544)

[9.7.3 Diagrama contextual 51](#_Toc517867545)

[9.7.4 Justificación de las decisiones 51](#_Toc517867546)

[10 Bibliografía 52](#_Toc517867547)

# Introducción y Objetivos

El objetivo de este documento es definir la estructura de una arquitectura para gestión de incidencias que pueda ser reutilizada. Aunque el sistema aquí descrito tiene funcionalidad propia, el objetivo es que pueda integrarse como parte de un sistema general de participación ciudadana.

La arquitectura aquí descrita forma parte de la práctica realizada por Alejandro Barrera para la convocatoria extraordinaria de junio de la asignatura Arquitectura del Software, impartida en el grado de Ingeniería Informática del Software, Escuela de Ingeniería Informática, Universidad de Oviedo por los profesores: Aquilino Adolfo Juan Fuente, Jose Emilio Labra Gayo, Juan Luis Mateo Cerdán y Herminio García González.

El sistema se ha descompuesto en cuatro partes: Loader para cargar datos de los ciudadanos del municipio, Agents, para que un ciudadano se dé de alta como participante y consultar si puede participar, InciManager que permite la introducción de nuevas incidencias en el sistema, y InciDashboard para poder consultar el estado actual y en tiempo real de las incidencias.

El estudiante deberá plantear y describir una arquitectura que se ajuste al problema planteado y realizar la implementación de un prototipo del sistema de participación ciudadana. Ya cuenta con la implementación de las partes Loader y Agents, implementados por los estudiantes del curso 2016-2017 y la actualización de ambas partes realizada en la convocatoria ordinaria de 2017-2018.

# Requisitos

La gestión de incidencias se hará siguiendo un esquema de cuatro partes:

* Loader: Carga la lista de usuarios del Ayuntamiento, por ejemplo, el padrón municipal.
* Agents: Permite chequear si un agente puede participar
* InciManager: Permite añadir nuevas incidencias al sistema. Y revisar las ya enviadas.
* InciDashboard: Permite consultar el estado de las incidencias. Y cambiar su estado.

## Loader

El Administrador del sistema debe poder introducir los datos de los agentes. Puede haber distintos tipos de agentes: personas físicas, entidades, sensores, etc. Cada tipo de usuario estará identificado por una palabra clave. Por ejemplo: Ciudadano, Entidad, Sensor…

La introducción de los datos se hará a partir de ficheros Excel. Los campos del fichero Excel son los siguientes:

* Nombre: (en el caso de personas, contendrá nombre y apellidos)
* Localización: (coordenadas geográficas del agente). Este valor es opcional para personar 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 del agente.

Además del fichero Excel, antes descrito, 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 es el nombre del tipo de usuario.

Durante la importación de estos datos se creará un usuario y una clave aleatoria que le permita acceder al sistema para comprobar que está dado de alta. El sistema generará unas cartas personalizadas que se enviarán a los correos electrónicos de los agentes. Este envío es realizado por el propio ayuntamiento y no forma parte de este sistema.

Si un usuario figura ya en el sistema, esta eventualidad debe ser identificada, registrada en el fichero de log y el usuario sólo podrá ser creado una vez. Si los datos del usuario no son los mismos, no se modificarán los datos en el sistema y se registrará esta eventualidad en fichero de log.

Una vez importando un fichero Excel conteniendo una lista de agentes, se emitirá un email para cada usuario comunicándole que ha sido añadido al Portal de Participación Ciudadana, su usuario y su clave de acceso.

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

Si el fichero contiene errores, se detectarían y se enviarían los datos a un fichero de LOG para su posterior tratamiento.

## Agents

Los agentes deben poder acceder al sistema para comprobar que han sido dados de alta, a partir de la información recibida por email (opcionalmente también por carta). Para ello se creará un servicio web muy simple que toma como parámetros codificados en una llamada POST el nombre de usuario y la clave,y devuelva información sobre los datos que figuran sobre el ciudadano en el sistema en caso correcto o informe del error en case incorrecto. Tanto los parámetros como la respuesta se enviarán en formato JSON.

El formato de la petición POST es el siguiente:

{“login”: usuario, “password”: password, “kind”: tipo de agente}

En caso de que la combinación sea correcta, se devolverá la siguiente información:

{“name”: Nombre, “location”: Coordenadas, “email”: Email, “id”: identificador, “kind”: tipo de usuario, “kindCode”: código numérico del tipo de usuario }

Para garantizar la integridad de los datos el campo “kindCode” es el código de usuario que se alberga en la base de datos y que puede coincidir o no con el fichero maestro usado por loader. Cada código es único y se relaciona inequívocamente con un único tipo de usuario.

Este sistema dispone de un cliente escrito en HTML5, CSS3 y JS para que los agentes puedan entrar al sistema, consultar su información y modificar algunos datos como el email o la contraseña.

## InciManager

Este sistema se encarga de tramitar las incidencias que serán enviadas por los agentes que estén dados de alta en el sistema y tengan permiso para enviar incidencias.

Cada incidencia puede contener la siguiente información:

* Nombre de usuario y contraseña
* 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)
* 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.

Si el nombre de usuario y la contraseña son incorrectos, 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 al sistema de mensajería Apache Kafka para que sea analizada.

Los sensores pueden estar enviando incidencias de forma continua, dependiendo de cómo estén configurados.

Las incidencias tendrán uno de los siguientes estados:

* Abierta
* En proceso
* Cerrada
* Anulada

También tendrán más información generada por el sistema, que será la persona/entidad asignada para resolver la incidencia, url para más información... Algunas incidencias pueden tener una caducidad.

Solo se almacenará un subconjunto de las incidencias enviadas (las enviadas por personas o entidades, o algunos valores específicos enviados por los sensores).

Los agentes podrán consultar las incidencias que han enviado y realizar un seguimiento de estas, utilizando el nombre de usuario y la clave asignadas previamente.

## 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 a través de Apache Kafka.

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.

Se ofrecerán visualizaciones gráficas de las incidencias en forma de gráficos.

# Metodología usada

Se va a realizar un estudio de arquitectura siguiendo el método de ADD (Atribute-Driven Design) (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>

# Identificación de *Stakeholders*

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

1. Alejandro Barrera Sánchez
2. Administrador del Sistema
3. Agentes
4. Responsables políticos del portal
5. Profesores de la asignatura
6. Operarios
7. Grupo de alumnos que realizaron la práctica en convocatoria ordinaria

Así pues, la lista de *stakeholders* queda:

| Código | Stakeholder | Intereses (Módulos) |
| --- | --- | --- |
| ST-01 | Alejandro Barrera Sánchez | Todos |
| ST-02 | Administrador del Sistema | Loader |
| ST-03 | Agentes | Agents, InciManager |
| ST-04 | Responsables políticos del portal | Agents, InciManager, InciDashboard |
| ST-05 | Profesores de la asignatura | Todos |
| ST-06 | Operarios | InciDashboard |
| ST-07 | G. Alumnos C. Ordinaria | Loader, Agents |

Tabla 1. Lista de Stakeholders e intereses

Posteriormente se pasa a describir en más detalle cada uno.

## Alejandro Barrera Sánchez

Se trata del Arquitecto y del desarrollador software que luego implementará el prototipo.

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 profesional.
* Utilización de tecnologías entre los distintos módulos para evitar incompatibilidades.

## Administrador del Sistema

Es la persona que carga los ficheros de datos. Y la que se va a ocupar de mantener los servidores en óptimas condiciones.

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 ciudadanos
* Ser capaz de depurar el proceso de carga en caso de errores

## Agentes

Son parte de los usuarios finales del sistema.

Entre sus objetivos están:

* Sencillez de acceso a los datos.
* Ser capaz de acceder desde su casa de una forma segura.
* Ser capaz de consultar el estado del sistema
* Ser capaz de cambiar su información en el sistema, por ejemplo, la clave.
* Ser capaz de registrar una incidencia.
* Ser capaz de ver las incidencias enviadas.

## Responsables políticos del portal

Está formado por el equipo que se encarga de explotar los resultados de la participación ciudadana en el portal. Entre sus objetivos están:

* Disponer de información sobre lo que ocurre en el portal tan pronto como sea posible.
* Utilizar tecnologías fáciles de usar e interoperables con otros sistemas.

## 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-driven design*)
* 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.

## Operarios

Serán los trabajadores del ayuntamiento que resolverán las incidencias haciendo uso del InciDashboard.

Entre sus objetivos están:

* Facilidad de aprendizaje y uso de la herramienta.
* Que la herramienta sea una utilidad a su día a día en el trabajo y no entorpezca o complique su trabajo.
* Disponer de datos útiles y sencillo de entender sobre las incidencias.
* Ser capaces de cambiar el estado de las incidencias.

## Grupo de alumnos que realizaron la práctica en convocatoria ordinaria

Para la realización de esta práctica y esta arquitectura me baso en parte de la arquitectura diseñada y código escrito durante las prácticas de la convocatoria de mayo realizada, aunque sin éxito, entre todo el grupo 5A. La arquitectura e implementación del módulo Agents y Loader quedan casi iguales a lo realizado durante la convocatoria ordinaria.

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.
* Que quede reconocido su trabajo en este proyecto.

# Atributos de calidad

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

* **Disponibilidad**
  + Disponibilidad del sistema un mínimo del 99% del tiempo.
  + Disponibilidad del módulo Agents, aunque el sistema Loader esté cargando los datos.
  + Tolerancia a fallos físicos (Discos duros estropeados, CPUs quemadas…).
* **Modificabilidad**
  + Facilidad de cambio de diversas partes de la aplicación: Cambiar el *parser* de entrada de listas de ciudadanos para ficheros con nuevos campos o incluso otro tipo de ficheros.
  + Facilidad de cambio de diversas partes de la aplicación: Añadir nueva información al fichero de *log*
  + Facilidad para modificar partes de la aplicación: otros formatos de salida para las cartas personales
  + Facilidad de cambio de diversas partes de la aplicación: Permitir el cambio de contraseña, permitir el cambio de *email*…
  + Facilidad para modificar partes de la aplicación: otros formatos, mediante negociación de contenido
  + Facilidad para modificar partes de la aplicación: cambiar la configuración y vista del cuadro de mandos
* **Rendimiento**
  + El rendimiento del proceso de carga de los ficheros es razonable.
  + La consulta de información de un usuario por el servicio web debe ser rápida.
  + Un agente debe ser capaz de comprobar su incidendia en el módulo InciManager, con un tiempo de retraso máximo de 15 segundos.
  + La incidencia debe aparecer en el dashboard con un máximo de 30 segundos de diferencia desde que el usuario la ha introducido a través del InciManager.
  + La información de sus datos debe estar disponible a través del módulo Agents justo después de que el usuario haya sido informado de sus claves de acceso.
  + Si fuese necesario, se debería poder añadir recursos al servidor.
* **Seguridad**
  + Garantizar la confidencialidad de los datos.
  + Garantizar que un usuario no identificado no pueda acceder a los datos de las incidencias.
  + Garantizar que sólo los operarios tienen acceso al dashboard.
  + Garantizar que en caso de un robo de datos los datos sensibles sean ilegibles por el atacante.
  + Obligar a los usuarios a mantener unas contraseñas y habitos seguros en el uso del sistema.
  + Ser inmune a SQLInjection.
* **Testabilidad**
  + Debe ser posible chequear automáticamente que los datos del fichero se cargan adecuadamente.
  + Debe ser posible chequear que el servicio Web se comporta de forma adecuada.
  + Debe ser posible comprobar el buen funcionamiento de los módulos independientemente.
* **Usabilidad**
  + El sistema de carga de datos debe poder ser usado por usuarios administradores de sistema familiarizados con herramientas tipo Unix.
  + Todos los clientes web del sistema deben contar con una accesibilidad mínima AA.
  + Curva de aprendizaje leve en los clientes web.
* **Interoperabilidad**
  + El presente sistema será usado por el Sistema de Participación Ciudadana, el cual delegará en el sistema actual la gestión de incidencias. El subsistema *Agents* deberá ser utilizado por un proceso automático para consultar el estado de los usuarios que quieran participar.
  + El servicio web integrado en Agents debe poder ser consultado desde cualquier lenguaje de programación y cualquier plataforma.
  + Los eventos producidos por el sistema InciManager deben ser poder consumidos por cualquier aplicación, independientemente de la plataforma en la que se ejecute y el lenguaje de programación.
  + El módulo InciDashboard puede consumir eventos producidos por cualquier aplicación, independientemente de la plataforma en la que se ejecute y el lenguaje de programación.
* **Simplicidad**
  + Los cuatro sub-sistemas deberán ser simples y fáciles de desarrollar.
* **Desplegabilidad**
  + El sistema debe ser fácilmente desplegable, especialmente en un servidor en la nube.
  + Se debe poder desplegar los módulos en máquinas físicas o lógicas distintas.
  + Se debe poder desplegar los módulos individualmente.
  + Se debe poder desplegar el sistema en distintos SO.

## Lista de atributos de calidad

| **Código** | **Descripción** | **Tipo de Atributo** | **Módulo afectado** |
| --- | --- | --- | --- |
| **AT001** | Disponibilidad del sistema para consultar datos el 99% del tiempo. | Disponibilidad | Agents, InciManager, InciDashboard |
| **AT002** | Facilidad de cambio de diversas partes de la aplicación: Cambiar el parser de entrada de listas de ciudadanos. | 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** | Facilidad de cambio de diversas partes de la aplicación: Permitir el cambio de contraseña, permitir el cambio de email, permitir el cambio de dirección postal. Permitir cambiar los campos JSON de entrada/salida | Modificabilidad | Agents |
| **AT006** | Facilidad para cambiar partes de la aplicación: procesar y devolver información en otros formatos mediante negociación de contenido. | 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 | Loader, Agents, InciManager e 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 usado por administradores de sistemas familiarizados con herramientas tipo Unix | Usabilidad | Loader |
| **AT012** | El servicio Web 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 | Loader, Agents, InciManager e InciDashboard |
| **AT014** | El sistema debe ser fácilmente desplegable | Desplegabilidad | Loader, Agents, InciManager e InciDashboard |
| **AT015** | El módulo Agents debe estar disponible, aunque el módulo Loader esté modificando los datos. | Disponibilidad | Agents y Loader |
| **AT016** | Facilidad para modificar partes de la aplicación: cambiar la configuración y vista del cuadro de mandos | Modificalidad | InciDashboard |
| **AT017** | Después de registrar su incidencia el agente debe poder consultarla en un tiempo razonable (no demasiado lento, pero tampoco crítico) | Rendimiento | InciManager |
| **AT018** | Después de registrarse una incidencia esta debe aparecer en el dashboard de los operarios en un tiempo razonable (no demasiado lento, pero tampoco crítico) | Rendimiento | InciManager e InciDashboard |
| **AT019** | Un agente puede consultar sus datos almacenados una vez tenga sus datos de acceso. | Rendimiento | Loader y Agents |
| **AT020** | Un usuario no identificado no pueda acceder a los datos de las incidencias. | Seguridad | InciDashboard |
| **AT021** | Sólo los operarios tienen acceso al dashboard | Seguridad | InciDashboard |
| **AT022** | En caso de un robo de datos los datos sensibles seran ilegibles por el atacante | Seguridad | Loader y InciManager |
| **AT023** | Obligar a los usuarios a mantener unas contraseñas y habitos seguros en el uso del sistema | Seguridad | Agents, InciManager e InciDashboard |
| **AT024** | Evitar la vulnerabilidad SQLInjection. | Seguridad | Agents, InciManager e InciDashboard |
| **AT025** | Debe ser posible comprobar el buen funcionamiento de los módulos independientemente. | Testabilidad | Loader, Agents, InciManager e InciDashboard |
| **AT026** | **BORRADO** |  |  |
| **AT027** | Todos los clientes web del sistema cuentan con una accesibilidad mínima AA. | Usabilidad | Agents, InciManager e InciDashboard |
| **AT028** | Todos los clientes web deben tener una curva de aprendizaje leve | Usabilidad | Agents, InciManager e InciDashboard |
| **AT029** | El servicio web integrado en Agents debe ser interoperable. | Interoperabilidad | Agents |
| **AT030** | Los eventos producidos por InciManager deben poder consumirse con cualquier aplicación. | Interoperabilidad | InciManager |
| **AT031** | InciDashboard puede consumir eventos de cualquier aplicación. | Interoperabilidad | InciDashboard |
| **AT032** | Se debe poder desplegar los módulos en máquinas físicas o lógicas distintas | Desplegabilidad | Agents, InciManager e InciDashboard |
| **AT033** | Se debe poder desplegar los módulos individualmente | Desplegabilidad | Agents, InciManager e InciDashboard |
| **AT034** | Se debe poder desplegar el sistema en distintos SO | Desplegabilidad | Agents, InciManager e InciDashboard |
| **AT035** | El sistema debe ser tolerante a fallos físicos | Disponibilidad | Loader, Agents, InciManager, InciDashboard, Kafka y BBDD |
| **AT036** | Si la carga de trabajo lo requiriese, debería ser fácil añadir más recursos a los servidores. | Rendimiento | Loader, Agents, InciManager, InciDashboard, Kafka y BBDD |

Tabla 2. Lista de atributos de calidad y tipos

## Atributos de calidad e Interesados

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** | **ST-07** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **AT001** | X |  | X | X | X | X | X |
| **AT002** | X | X |  |  | X |  | X |
| **AT003** | X | X |  |  | X |  | X |
| **AT004** | X | X |  |  | X |  | X |
| **AT005** | X |  | X |  | X |  | X |
| **AT006** | X |  | X | X | X |  | X |
| **AT007** | X | X |  |  | X |  | X |
| **AT008** | X | X |  |  | X | X | X |
| **AT009** | X |  |  |  | X |  | X |
| **AT010** | X |  | X | X | X |  | X |
| **AT011** | X | X |  |  | X |  | X |
| **AT012** | X |  |  | X | X |  | X |
| **AT013** | X |  |  | X | X |  | X |
| **AT014** | X | X |  |  | X |  | X |
| **AT015** | X | X | X | X | X |  | X |
| **AT016** | X |  |  |  | X | X |  |
| **AT017** | X |  | X | X | X |  |  |
| **AT018** | X |  |  | X | X | X |  |
| **AT019** | X |  | X | X | X |  | X |
| **AT020** | X |  | X |  | X | X |  |
| **AT021** | X |  |  |  | X | X |  |
| **AT022** | X | X | X | X | X |  | X |
| **AT023** | X |  | X | X | X |  | X |
| **AT024** | X |  | X | X | X |  | X |
| **AT025** | X | X |  |  | X |  | X |
| **AT027** | X |  | X | X | X | X | X |
| **AT028** | X |  | X |  | X | X | X |
| **AT029** | X |  |  |  | X |  | X |
| **AT030** | X |  |  | X | X |  |  |
| **AT031** | X |  |  | X | X |  |  |
| **AT032** | X |  |  | X | X |  | X |
| **AT033** | X |  |  |  | X |  | X |
| **AT034** | X |  |  |  | X |  | X |
| **AT035** | X | X |  |  | X |  |  |
| **AT036** | X | X |  |  | X |  |  |

Tabla 3. Lista de intereses de los *stakeholders*

# Restricciones

Para realizar esta aplicación existen las siguientes restricciones

## Restricciones técnicas

| **Código** | **Restricción** | **Motivación** |
| --- | --- | --- |
| **TC001** | El lenguaje de programación será Java | Se asume que el equipo de desarrollo (ST-01, ST-07) tiene conocimientos de Java |
| **TC002** | Se utilizará una o varias bases de datos relacionales para almacenar los datos | Se asume que el equipo de desarrollo (ST-01, ST-07) 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. Además de ser interoperable. |
| **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 personalizadas será, al menos, en texto plano | Con el fin de facilitar la implementación se propone generar cartas personalizadas mediante texto plano. El equipo de desarrollo puede implementar otros formatos, como PDF o Word. |
| **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** | Los módulos InciManager e InciDashboard se comunicarán a través de Apache Kafka | Apache Kafka es un sistema de almacenamiento publicador/subscriptor distribuido, particionado y replicado. Existen numerosos ejemplos y material de ayuda para facilitar su aprendizaje. Apache Kafka además ofrece interoperabilidad. |
| **TC009** | Los módulos InciManager e InciDashboard se implementarán en el framework Spring Boot | El framework Spring Boot proporciona mucha funcionalidad ya implementada y probada que ayudará a la implementación del prototipo. También ofrece integración con Kafka y permite consumir servicios web fácilmente. |
| **TC010** | El módulo InciManager enviará a través de Kafka las incidencias en formato JSON | JSON, acrónimo de JavaScript Object Notation, es un formato de texto ligero para el intercambio de datos. |

Tabla 4. Restricciones técnicas

## Restricciones organizativas

| **Código** | **Restricción** | **Motivación** |
| --- | --- | --- |
| **OC001** | Cada sub-sistema será implementado individualmente. | El alumno trabajará solo con el fin de que el estudiante pueda aprender a plantear e implementar una arquitectura. |
| **OC002** | La estructura de la base de datos será la misma para Loader y Agents | El pegamento entre los 2 sub-sistemas es la base de datos, cuya estructura debe ser acordada por los 2 equipos. |
| **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 |
| **OC004** | La estructura de la base de datos será la misma para InciManager e InciDashboard | Los sub-sistemas InciManager e InciDashboard compartirán BBDD para simplificar el desarrollo del sistema. |

Tabla 5. Restricciones organizativas

# Ámbito del sistema y contexto

Para describir la solución se utilizarán diagramas contextuales y texto.

La aplicación está partida en cuatro módulos:

* Loader: Se encarga de la carga de los ficheros. Utiliza el estilo Batch
* Agents: Se encarga de las comprobaciones de los participantes. Utiliza el estilo micro-servicios.
* InciManager: Se encargar de recibir, procesar y almacenar las incidencias. Utiliza el estilo MVC.
* InciDashboard: Se encarga de mostrar las incidencias en un cuadro de mandos en tiempo real. Utiliza el estilo Kappa.

Los módulos Loader y Agents se integran usando el estilo arquitectónico de datos compartidos.

Los módulos InciManager e InciDashboard se integran mediante el estilo de integración Mensajería y datos compartidos.

También, los módulos InciManager e InciDashboard se integrarán con el módulo Agents consumiendo el servicio web REST que este contiene.

En el diagrama de contexto de la Figura 1, se muestran las principales interfaces de cada sistema. El subsistema AgentsDatabase es común a Loader y Agents, se utilizará un repositorio de tipo SQL relacional.

Es sistema de mensajería entre InciManager e InciDashboard a usar es el de Apache Kafka, el cual nos viene como restricción. Hay que acordar la configuración de Kafka y el formato de mensajes a usar, que será JSON.

El subsistema InciDataBase es común a InciManager y InciDashboard, se utilizará un repositorio de tipo SQL relacional.

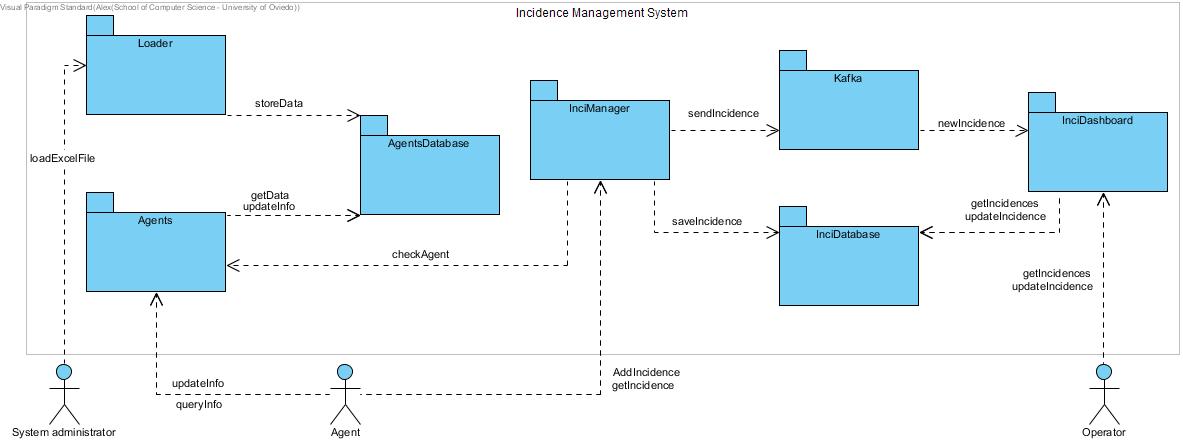


Figura 1. Contexto de negocio del sistema

A continuación, se incluye un diagrama BPMN que define el proceso completo de todos los subsistemas.

A destacar:

* Hay una base de datos común entre Agents y Loader y otra entre InciManager e InciDashboard.
* Los datos intercambiados usan formato JSON.
* Los procesos de Loader y de Agents son asíncronos.
* InciManager e InciDashboard usan Kafka para comunicarse.
* InciDashboard actualizarán la vista del cliente en tiempo real.

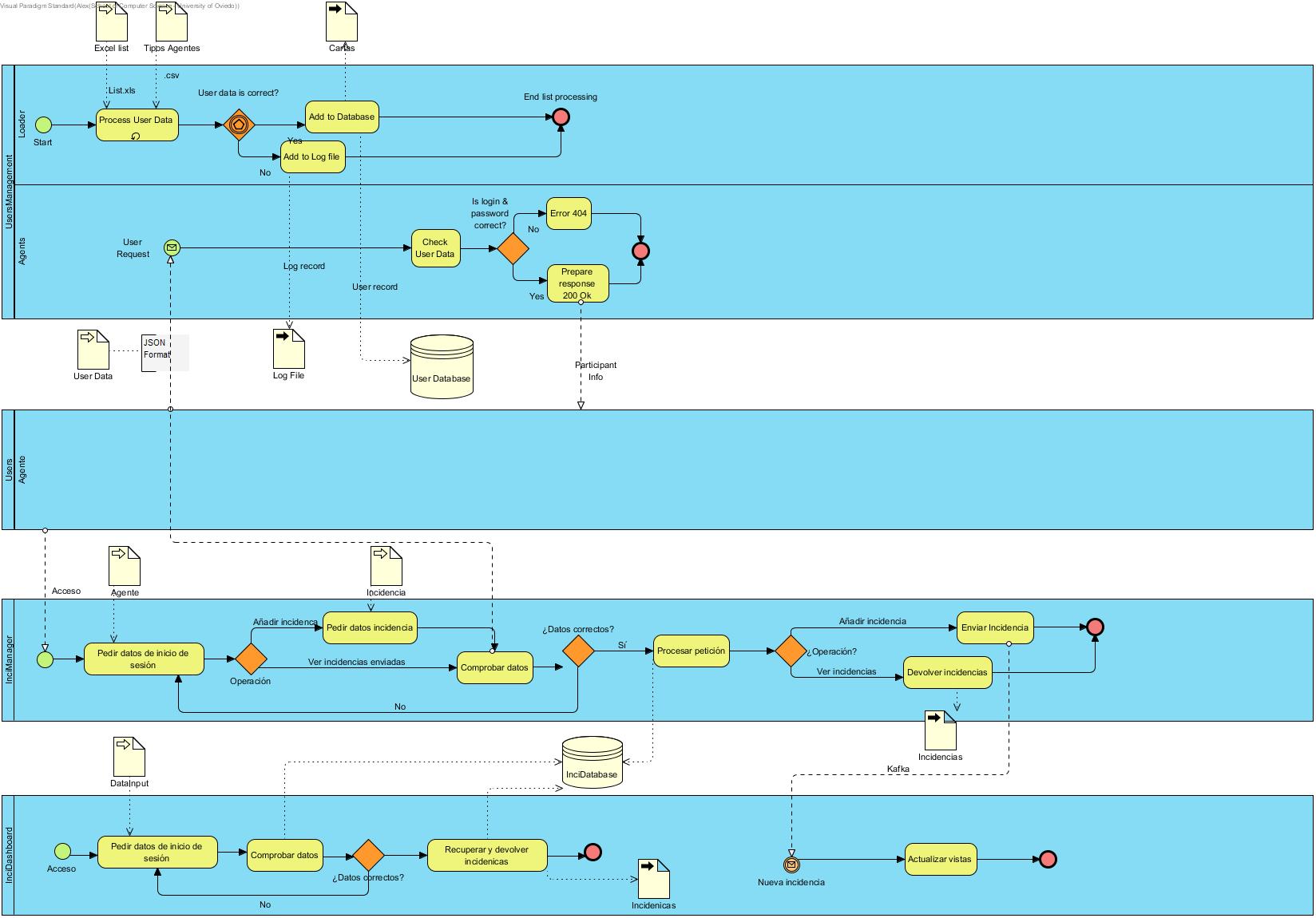


Figura 2. Diagrama BPMN

# 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 | Agente | Realiza una petición de información sobre sus datos | Tiempo de ejecución | Agents | El agente recibe la respuesta adecuada | La información es recibida en menos de 15 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 | Agents y DBManagement | Se incluye el nuevo formato con cambios mínimos en el código | Menos de un día de trabajo | **AT004** |
| 5 | Desarrollador | Se introduce la opción de cambio de información de los participantes | Desarrollo | Agents y DBManagement | La información se cambia adecuadamente o se genera el error identificando el problema | Menos de un día de trabajo | **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 ciudadanos | **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 | Participantes | Accede a la aplicación | Tiempo de ejecución | Agents | Cada ciudadano puede acceder a sus datos, pero no a los datos de otros ciudadanos | El acceso a los datos se permite solamente cuando la información de email/contraseña son correctas. | **AT009** |
| 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 ciudadano tiene menos información que la requerida. | **AT010** |
| 11 | Administrador del Sistema | Carga un fichero Excel en la base de datos | Tiempo de ejecución | Parser, DBUpdate y ReportWriter | El proceso de carga 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 Participación Ciudadana | Accede al servicio Web | Tiempo de ejecución | Agents | El sistema de Participación Ciudadana 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 | Los desarrolladores pueden implementar el sistema | El sistema puede implementarse en 2 semanas | **AT013** |
| 14 | Administrador del Sistema | Despliega el sistema | Despliegue | Loader, Agents | El sistema es desplegado en un entorno de producción | El sistema puede desplegarse en menos de una hora | **AT014** |
| 15 | Agente | Realiza una petición de información sobre sus datos | Loader está guardando información en la BBDD | Loader, Agents | El agente recibe la respuesta adecuada | La información es recibida en menos de 20 seg. | **AT015** |
| 16 | Desarrollador | El desarrolladles decide implementar una nueva vista | Desarrollo | InciDashboard | El desarrollador es capaz de hacer una nueva vista | Menos de un día | **AT016** |
| 17 | Agente | El agente registra una incidencia y la consulta. | Tiempo de ejecución | InciManager | El sistema le indica el estado de la incidencia. | Pasados menos de 15s desde el registro de la incidencia el sistema la devuelve en menos de 15s. | **AT017** |
| 18 | Agente | El agente registra una nueva incidencia. | Tiempo de ejecución | InciManager, InciDashboard | El sistema muestra la incidencia en todos los dashboards | Menos de 30s | **AT018** |
| 19 | Agente | El Agente recibe sus claves de acceso y realiza una petición de su información. | Tiempo de ejecución | Loader, Agents | El agente recibe la respuesta adecuada | La información es recibida en menos de 15 seg. | **AT019** |
| 20 | Usuario no operario | Intenta acceder al dashboard mediante url | Tiempo de ejecución | InciDashboard | El sistema bloquea la petición. | El usuario recibe la página de login | **AT020** |
| 21 | Agente | Intenta loguearse con sus datos en el dashboard | Tiempo de ejecución | InciDashboard | El sistema devuelve un error de login | El usuario recibe la página de login | **AT021** |
| 22 | Usuario malintencionado | Accede a la BBDD | Cualquiera | Base de datos, Loader | Los datos sensibles están hasheados o encriptados | El usuario es incapaz de leer los datos | **AT022** |
| 23 | Agente | Cambia su contraseña por una considerada como insegura | Tiempo de ejecución | Agents | El sistema no le deja cambiar la contraseña. | El usuario recibe un aviso. | **AT023** |
| 24 | Usuario Malintencionado | Inyecta código SQL | Tiempo de ejecución | Agents, InciManager e InciDashboard | El sistema fuerza a usar lo introducido por el usuario como datos y no interpreta el código. | El código inyectado no se ejecuta. | **AT024** |
| 25 | Desarrollador | Ejecuta los tests de un módulo | Desarrollo | Loader, Agents, InciManager, InciDashboard | Los tests de un módulo no depende de otro, se ejecutan sin problemas. | Los tests pasan satisfactoriamente. | **AT025** |
| 26 | **BORRADO** | - | - | - | - | - | **AT026** |
| 27 | Usuario con discapacidad | Navega por los clientes web | Tiempo de ejecución | Agent, InciManager, InciDashboard | Puede navegar sin problemas. | Puede navegar sin grandes problemas. | **AT027** |
| 28 | Usuario | Un nuevo usuario comienza a usar nuestro sistema | Tiempo de ejecución | Agent, InciManager, InciDashboard | Consigue navegar sin mucha dificultad | En menos de media hora de uso ya es capaz de manejar el sistema sin problemas | **AT028** |
| 29 | Aplicación de terceros | Aplicación de terceros realiza una petición a nuesto servicio web | Tiempo de ejecución | Agents | El sistema responde independientemente del lenguaje del otro programa | Menos de 15s. | **AT029** |
| 30 | Aplicación de terceros | La aplicación se subscribe al bus de mensajería | Tiempo de ejecución | InciManagery Kafka | Independientemente del lenguaje, la aplicación puede subscribirse y consumir los eventos. | En menos de un día la aplicación se integra en el sistema | **AT030** |
| 31 | Aplicación de terceros | La aplicación publica en el bus de mensajería | Tiempo de ejecución | InciDashboard y Kafka | Si la aplicación de terceros publica en el formato correcto, InciDashboard puede consumir sus eventos. | En menos de un día la aplicación se integra en el sistema | **AT031** |
| 32 | Administrador del sistema | Despliega el sistema en varias máquinas | Despliegue | Agents, InciManager e InciDashboard | El sistema es desplegado en un entorno de producción | Menos de 2 horas | **AT032** |
| 33 | Desarrollador | Modifica un módulo y lo despliega | Tiempo de ejecución | Agents, InciManager e InciDashboard | El sistema vuelve a funcionar sin problemas | Menos de 15 minutos. | **AT033** |
| 34 | Administrador del sistema | Cambia el SO del servidor | Tiempo de ejecución | Agents, InciManager e InciDashboard | El sistema es desplegado en un entorno de producción | Menos de 1 hora | **AT034** |
| 35 | Fallo en el disco duro. | El disco duro se estropea. | Tiempo de ejecución. | Todos | El sistema no se viene abajo. | La ejecución no se ve afectada. | **AT035** |
| 36 | Usuarios | Gran carga de trabajo. | Tiempo de ejecución | Todos | El administrador del sistema asigna más recursos a los servidores. | Menos de 2 horas. | **AT036** |
| 37 | ESCENARIO OVBIO | - | - | - | - | - | **TC001** |
| 38 | ESCENARIO OVBIO | - | - | - | - | - | **TC002** |
| 39 | Programa | Realiza una petición en formato JSON a Agents | Tiempo de ejecución | Agents | El sistema responderá de manera acorde | Menos de 1 minuto | **TC003** |
| 40 | ESCENARIO OVBIO | - | - | - | - | - | **TC004** |
| 41 | Administrador | Ejecuta Loader | Tiempo de ejecución | Loader | El sistema generará cartas en texto plano | < 1 segundo por cada 10 ciudadanos | **TC005** |
| 42 | ESCENARIO OVBIO | - | - | - | - | - | **TC006** |
| 43 | ESCENARIO OVBIO | - | - | - | - | - | **TC007** |
| 44 | ESCENARIO OVBIO | - | - | - | - | - | **TC008** |
| 45 | ESCENARIO OVBIO | - | - | - | - | - | **TC009** |
| 46 | InciManager | Nueva incidencia. Envio por Kafka en formato JSON | Tiempo de ejecución | InciManager e InciDashboard | El sistema muestra la incidencia a los operarios | Menos de un minuto | **TC010** |
| 47 | ESCENARIO OVBIO | - | - | - | - | - | **OC001** |
| 48 | ESCENARIO OVBIO | - | - | - | - | - | **OC002** |
| 49 | ESCENARIO OVBIO | - | - | - | - | - | **OC003** |
| 50 | ESCENARIO OVBIO | - | - | - | - | - | **OC004** |

Tabla 6. Lista de escenarios de calidad

# 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 | Atributos de calidad | Escenarios |
| Context | ST-01, ST-02, ST-03, ST-04, ST-05, ST-06, ST-07 | AT013, AT025, AT030, AT031, AT032, AT033, AT034, TC001, TC002, TC006, TC008, 0C001, 0C002, 0C003, 0C004 | 13, 25, 30, 31, 32, 33, 34, 37, 38, 42, 44, 47, 48, 49, 50 |
| Loader | ST-01, ST-02, ST-04, ST-05, ST-07 | AT002, AT003, AT007, AT008, AT010, AT011, AT013, AT015, AT019, AT022, TC004, TC005 | 2, 3, 7, 8, 10, 11, 14, 15, 19, 22, 40, 41 |
| Agents | ST-01, ST-03, ST-04, ST-05, ST-07 | AT001, AT004, AT005, AT006, AT008, AT009, AT012, AT013, AT014, AT015, AT019, AT023, AT024, AT027, AT028, AT029, TC003, TC007 | 1, 4, 5, 6, 8, 9, 12, 13, 14, 15, 19, 23, 24, 27, 28, 29, 39, 43 |
| InciManager | ST-01, ST-03, ST-04, ST-05 | AT017, AT018, AT024, AT027, AT028, AT030, TC009, TC010 | 17, 18, 24, 27, 28, 30, 45, 46 |
| InciDashboard | ST-01, ST-04, ST-05, ST-06 | AT016, AT018, AT020, AT021, AT024, AT027, AT028, AT031, TC009, TC010 | 16, 18, 20, 21, 24, 27, 28, 31, 45, 46 |
| Paquetes | ST-01, ST-02, ST-05, ST-07 | AT004, AT006, AT013 | 4,6,13 |
| Despliegue | ST-01, ST-02, ST-05 | AT014, AT030, AT031, AT032, AT033, AT034, AT035, AT036 | 14,30,31,32,33,34, 35, 36 |

## Contexto

La vista de sistema describe los cuatro subsistemas en interacción, así como sus interfaces y las 2 bases de datos.

### Presentación principal

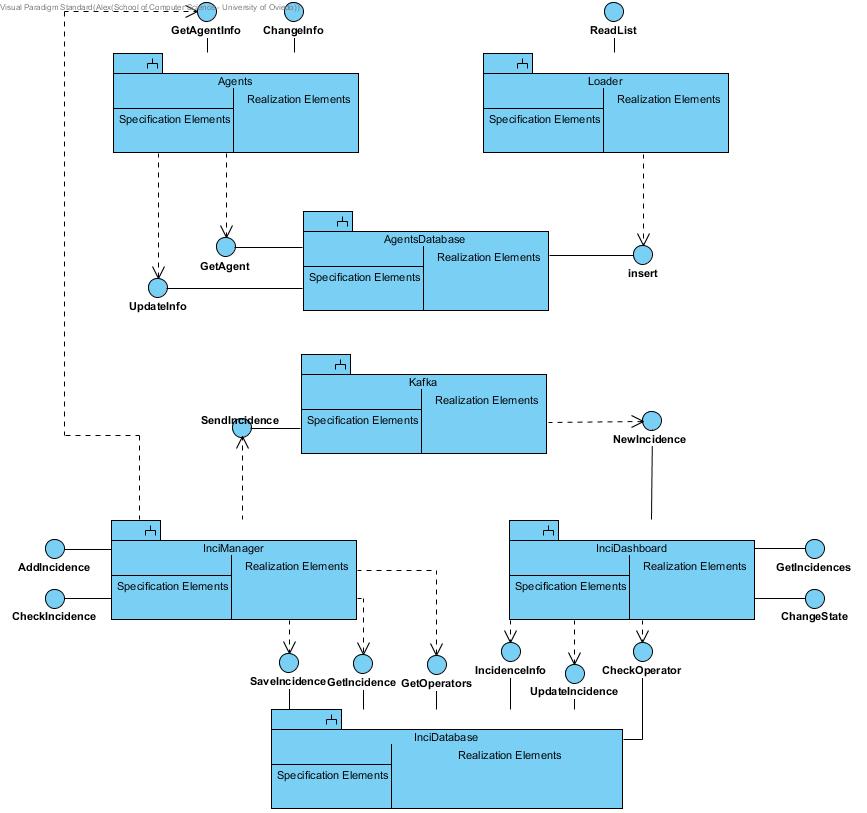


Figura 3. Context view

### Catálogo de elementos

#### Elementos

|  |  |
| --- | --- |
| Elemento | Propiedades |
| Loader | Se encarga de la introducción de las listas de ciudadanos en el sistema. Lee un fichero con los datos de los ciudadanos. 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. |
| AgentsDatabase | Este módulo encapsula los accesos a la base de datos referidos a los Agentes. |
| InciManager | Es el módulo usado por los agentes para enviar incidencias y les permite, además, seguir el estado de sus incidencias enviadas. |
| InciDashboard | Este módulo ofrece en tiempo real las vistas de las incidencias a los operarios y les permite cambiar su estado. |
| Kafka | Módulo que se encarga de enviar incidencias desde el InciManager a InciDashboard, en tiempo real. |
| InciDatabase | Este módulo encapsula los accesos a la base de datos referidos a las incidencias y operarios. |

#### 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.

Posteriormente se envían a la base de datos a través de la interfaz *Insert* del módulo Agents*Database*.

El módulo *Agents* permite al usuario entrar en sesión a través del servicio web *GetAgentInfo* para comprobar sus datos. Para ello, *Agents* pide los datos al módulo *AgentsDatabase* a través de la interfaz *GetAgent*. Además, a través de *ChangeInfo,* permite al usuario cambiar su clave u otros datos, para este fin, el módulo *Agents* solicita al módulo *DataBase* el cambio de clave a través de la clase *UpdateInfo*.

El módulo InciManager permite al usuario insertar incidencias en el sistema a través de la interface AddIncidence, el módulo envía todas las incidencias al módulo Kafka a través de la interface SendIncidence de este último. También las guarda en una base de datos a través de la interface SaveIncidence del módulo InciDatabase.

También, el módulo InciManager permite al agente comprobar el estado de las incidencias enviadas, para eso se nutre de la interface GetIncidence para recuperarlas de la base de datos.

Posteriormente el módulo Kafka envía las incidencias a InciDashboard a través de la interface NewIncidence para que este las procese en tiempo real.

InciDashboard permite a los operarios observar en tiempo real las incidencias que envían los agentes y cambiar su estado, para recuperar las incidencias y actualizarlas el módulo se sirve de la interface IncidencesInfo y UpdateIncidence respectivamente del módulo InciDatabase. También usa la interface CheckOperator, del módulo InciDatabase, para permitir loguearse a los operadores.

#### Interfaces / Puertos

##### Loader

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

##### Agents

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Tipo | Tecnología | Propiedades |
| GetAgenttInfo | Interface | Servicio Web | Este servicio se invocará a través de una petición HTTP |
| ChangeInfo | Interface | Servicio Web | Este servicio se invocará a través de una petición HTTP |

##### AgentsDataBase

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Tipo | Tecnología | Propiedades |
| GetAgent | Interface | Acceso a BBDD | Devuelve los datos de un ciudadano. |
| Insert | Interface | Acceso a BBDD | Inserta en la base de datos los datos de un ciudadano, incluida su contraseña |
| UpdateInfo | Interface | Acceso a BBDD | Actualiza la clave del usuario en la base de datos. |

##### InciManager

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Tipo | Tecnología | Propiedades |
| AddIncidence | Interface | Servicio Web | Permite añadir una nueva incidencia al sistema. |
| CheckIncidence | Interface | Servicio Web | Permite consultar el estado de una incidencia. |

##### InciDashboard

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Tipo | Tecnología | Propiedades |
| GetIncidences | Interface | Servicio Web | Permite consultar, en tiempo real, las incidencias que hay registradas en el sistema. |
| ChangeState | Interface | Servicio Web | Permite cambiar el estado de las incidencias. |
| NewIncidence | Interface | Invocación a método | Permite añadir una nueva incidencia en tiempo real. |

##### Kafka

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Tipo | Tecnología | Propiedades |
| SendIncidence | Interface | Invocación a método | Permite enviar una incidencia por el bus de mensajería. |

##### InciDatabase

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Tipo | Tecnología | Propiedades |
| SaveIncidence | Interface | Acceso a BBDD | Guarda una incidencia en la base de datos. |
| GetIncidence | Interface | Acceso a BBDD | Recupera una incidencia de la base de datos. |
| GetOperators | Interface | Acceso a BBDD | Recupera todos los operadores de la base de datos. |
| IncidenceInfo | Interface | Acceso a BBDD | Recupera una incidencia de la base de datos. |
| UpdateIncidence | Interface | Acceso a BBDD | Actualiza una incidencia en la base de datos. |
| CheckOperator | Interface | Acceso a BBDD | Comprueba que un operador esté en la base de datos. |

#### Comportamiento

##### Loader

Ver el párrafo 9.2.2.3.4.

Además, puede hacer las siguientes opciones:

* El subsistema que cree los emails se implementará de forma que permita emails o cartas en el futuro en diferentes formatos (Word, ODT, PDF, RTF, ETC.).
* Si el fichero viniera con errores, se detectarían y se enviarían los datos a un fichero de LOG para su posterior tratamiento.
* El *parser* de los datos de entrada debe ser configurable para permitir datos en diferentes formatos (Excel, TXT, etc.)

##### 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.

##### AgentsDatabase

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

##### InciManager

Permite a los agentes añadir nuevas incidencias al sistema, a través de un cliente WEB que ofrecerá este subsistema o a través de peticiones HTTP. Sólo permite enviar incidencias a agentes identificados. Después de validar la información y tomas decisiones las incidencias son guardas en la base de datos y enviadas a través de Kafka.

##### InciDashboard

Permite a los operarios consultar las incidencias en tiempo real y cambiar el estado de estas, todo esto a través de un servicio web REST. Ofrecerá un cliente web para los operarios. Se comunicará con la base de datos para consultar el histórico de incidencias y recibirá incidencias en tiempo real a través de Kafka.

##### InciDatabase

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

##### Kafka

Este módulo encapsulará las operaciones de comunicación entre los módulos InciManager e InciDashboard y la tecnología Apache Kafka.

### Diagrama contextual

Ver 9.1

### Justificación de las decisiones

|  |  |  |
| --- | --- | --- |
| Escenario | Atributos de calidad | Justificación |
| 13 | AT013 | El encapsulamiento de las funcionalidades entre los distintos subsistemas y el bajo acoplamiento de estos, hace posible que el sistema pueda desarrollarse de manera sencilla. |
| 25 | AT025 | El encapsulamiento de las funcionalidades y el bajo acoplamiento hace que sea posible hacer y ejecutar tests para probar gran parte de la funcionalidad de cada módulo de manera separada. |
| 30 | AT030 | El uso del subsistema Kafka hace que los módulos sean interoperables con otros módulos sin hacer falta que estos se conozcan. |
| 31 | AT031 | El uso del subsistema Kafka hace que los módulos sean interoperables con otros módulos sin hacer falta que estos se conozcan. |
| 32 | AT032 | El diseño de los módulos y las tecnologías elegidas (BBDD, Kafka, REST) permiten que los módulos se desplieguen independientemente y se comuniquen entre ellos. |
| 33 | AT033 | Al haber separado los módulos en subsistemas es posible actualizar uno y redesplegarlo sin modificar el resto. |
| 34 | AT034 | Al haber elegido Java como plataforma de desarrollo, esta nos proporciona independencia con el SO. |
| 37 | AT037 | Se va a desarrollar en el lenguaje multiplataforma Java. |
| 38 | AT038 | Se utilizan dos bases de datos SQL relacional, InciDatabase y AgentsDatabase. |
| 42 | AT042 | Se utilizará el framework de pruebas Junit. |
| 47 | AT047 | Los subsistemas pueden ser implementados individualmente y paralelamente. |
| 48 | AT048 | Se ha elegido usar la misma BBDD para Agents y Loader. |
| 49 | AT049 | Se usará el control de versiones git hosteado en GitHub. |
| 50 | AT050 | Se ha elegido usar la misma BBDD para InciManager e InciDashboard. |

## Loader

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

### Presentación principal

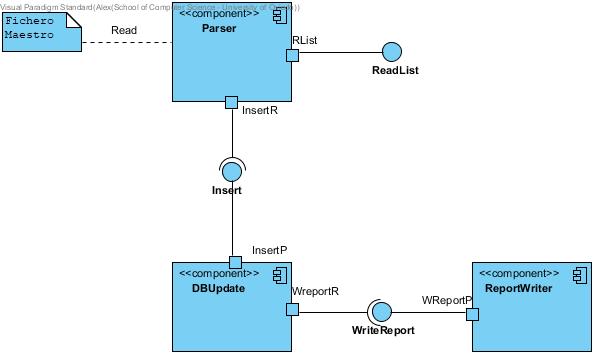


Figura 4. Vista Loader

### Catálogo de elementos

#### Elementos

| Elemento | Propiedades |
| --- | --- |
| Parser | Lee los datos de entrada en Excel y los transforma en un contenedor de objetos que puede ser recorrido para su inserción en la base de datos.  También crea el *identificador/password* del agente y el email 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. |
| DBUpdate | Encapsula todas las operaciones de base de datos usando interfaces para permitir el acceso a la base de datos. |
| ReportWritter | Recibe cadenas de información con los datos del usuario que fue imposible de dar 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. |

#### Relaciones

El componente *Parser* recibe el fichero de entrada en *Excel* y un fichero txt, que contiene la relación código y tipo de agente. Luego, mediante un *parser,* convierte éste en objetos. Añade a estos objetos el email y el *password*, y lo añade a la base de datos utilizando el componente *DBUpdate*.

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

#### Interfaces / Puertos

##### Parser

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Tipo | Tecnología | Propiedades |
| ReadList | Interface | Invocación a Métodos | Lee el fichero de *Excel* con los datos de una lista de agentes. |
| RList | Port |  | Crea los subcomponentes del *parser* necesarios para procesar el fichero de entrada. |
| Insert | Interface (Requerida) | Invocación a Métodos | Llama a un método del componente *DBUpdate* para hacer la inserción en la base de datos. |
| InserR | Port |  | Verifica los datos y crea el objeto a enviar a *DBUpdate*. |

##### DBUpdate

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Tipo | Tecnología | Propiedades |
| Insert | Interface | Invocación a Métodos | Recibe un objeto con la información para insertar en la base de datos. |
| InsertP | Port |  | Verifica los datos de entrada y si falta algún atributo obligatorio genera el correspondiente error. |
| WriteReport | Interface (Requerida) | Invocación a Métodos | Llama a un método del componente *ReportWriter* para escribir una línea o registro en el fichero de *log*. |
| WreportR | Port |  | Verifica los datos a escribir. |

##### ReportWriter

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Tipo | Tecnología | Propiedades |
| WriteReport | Interface | Invocación a Métodos | Recibe los datos para escribir en el fichero de *log*. |
| WreportP | Port |  | Añade a los datos la hora y la fecha. |

##### 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 (String)
* Localización (Location) opcional
* Email (String con un formato acorde a las convenciones de correo electrónico)
* Identificador (String formado por dígitos y letras)
* Tipo (int)

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

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

##### DBUpdate

Actualiza la base de datos. Ver 9.1.2.4.3.

##### ReportWriter

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

* Fecha
* Hora
* Fichero Excel de procedencia
* Descripción del error (con toda la información necesaria)

### Diagrama contextual

Ver 9.1.

### Justificación de las decisiones

Las decisiones que han llevado a este diseño son:

|  |  |  |
| --- | --- | --- |
| Escenario | Atributos de calidad | Justificación |
| 2 | AT002 | El acceso al *parser* mediante un patrón *Adapter* garantiza un cambio rápido de *parser* sin tocar el código ya realizado en otras partes de la aplicación. |
| 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. |
| 7 | AT007 | La utilización de una base de datos relacional ofrecerá un acceso eficiente a la información de los usuarios |
| 8 | AT008 | Utilizar una base de datos con características de seguridad habilitadas podrá garantizar que los datos están aislados de accesos indebidos. El envío de la contraseña d manera individualizada evita que la información pueda ser leída por otros ciudadanos. |
| 9 | AT009 | 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 | 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 |
| 11 | AT011 | La aplicación contará con una ayuda donde las opciones sean simples y estén bien explicadas. El sistema se testeará para verificar que siempre se comporta de una manera correcta. |
| 14 | AT014 | Una aplicación *batch* independiente puede ser ejecutada directamente sin ninguna necesidad especial para su despliegue |
| 15 | AT015 | El uso de una base de datos SQL Relacional nos ofrece acceso concurrente. |
| 19 | AT019 | El sistema terminará y enviará las claves de acceso cuando ya haya cargado los datos en la base de datos relacional, esto permite que el usuario pueda acceder en cuanto los ficheros se hayan generado. |
| 22 | AT022 | Se utiliza un algoritmo hash para hashear la contraseña. El uso de algoritmos modernos impide que a partir del hash se pueda conocer la clave de usuario. |
| 40 | AT040 | El sistema espera los datos de entrada en formato Excel. |
| 41 | AT041 | El formato de salida será texto plano. |

## Agents

### Presentación principal

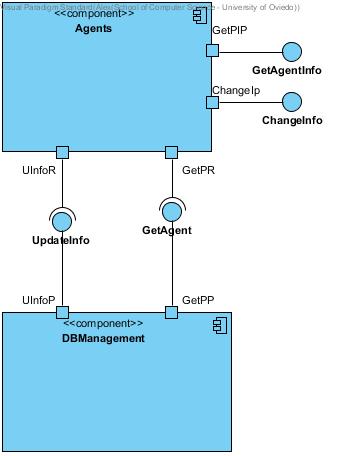


Figura 5. Vista de Agents

### Catálogo de elementos

#### Elementos

|  |  |
| --- | --- |
| Elemento | Propiedades |
| Agents | Se accede a través de dos servicios web: *GetAgent*, que permite al usuario acceder a sus datos en el sistema y (opcional) *ChangeInfo* que permite al usuario cambiar su clave de acceso u otros datos. |
| DBManagement | Se accede a través de dos interfaces: GetAgent, que devuelve los datos de un agente en la base de datos y (opcional) *UpdateInfo*, para actualizar un cambio de clave en la base de datos. |

#### Relaciones

El Sistema de Participación Ciudadana invoca *Agents* utilizando una llamada a un servicio web que es procesada por *GetParticipantInfo* (enviado *email/contraseña*) y éste accede a los datos encapsulados en DBManagement mediante la interface *GetParticipant*. Si la combinación *email/contraseña* es correcta se devuelve la información del usuario en formato JSON.

El usuario (agente) accede a *Agents* de forma manual al servicio web *ChangeInfo* enviado *usuario/password/newPasswod*) y éste llama a la interface *UpdateInfo* para modificar la clave a través del componente *DBManagement*.

#### Interfaces / Puertos

##### Agents

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Tipo | Tecnología | Propiedades |
| GetAgentInfo | Interface | Servicio Web | Permite el acceso a los datos de un agente mediante la combinación *email/contraseña*. |
| GetPIP | Port |  | Hace la validación del usuario antes de pedir los datos. |
| ChangeInfo | Interface | Servicio Web | Permite el cambio de clave u otra información a un ciudadano mediante una combinación de datos: *email/contraseña/nuevaContraseña*. |
| ChangeIP | Port |  | Hace la validación del usuario antes de solicitar el cambio de clave. |
| UpdateInfo | Interface (Requerida) | Invocación a Métodos | Solicita el cambio de clave para el usuario. |
| UInfoR | Port |  | -- |
| GetAgent | Interface (Requerida) | Invocación a Métodos | Solicita la información para el usuario. |
| GetPR | Port |  | -- |

##### DBManagement

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Tipo | Tecnología | Propiedades |
| UpdateInfo | Interface | Invocación a Métodos | Permite la solicitud de cambio de contraseña u otra información para el usuario. |
| UInfoP | Port |  | -- |
| GetAgent | Interface | Invocación a Métodos | Permite la solicitud la de información para el usuario. |
| GetPP | Port |  | -- |

#### Comportamiento

##### 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:

<URIServicioWeb>/user

Donde <URIServicioWeb> representa la URI en la que está desplegado el servicio Web. La petición POST contiene datos JSON con la siguiente estructura:

{"login": email, "password": password, "kind": tipo de agente}

En caso de que la combinación (email, password, tipò) aparezca en la base de datos, la respuesta será 200 OK con el cuerpo JSON de la forma:

{

"name": Nombre,

"location": Coordenadas (opcional),

"ID": Número documento identificativo,

"email": email

"kind": Tipo de usuario

"kindCode": Código numérico del tipo de usuario

}

En caso de que la combinación (email, password, tipo de agente) no aparezca, la respuesta será “404 Not found”.

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

El servicio Web permite a los usuarios cambiar su password.

##### DBManagement

Encapsula todos los accesos a la base de datos.

### Diagrama contextual

Ver 9.1.

### 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. |
| 4 | AT004 | La encapsulación de las características del modelo que afectan a la base durante el desarrollo y la utilización de un framework basado en MVC facilitará el desarrollo de nuevas funcionalidades como las vistas basadas en HTML o el cambio de clave o nuevos servicios de los usuarios. |
| 5 | AT005 | La encapsulación de la funcionalidad y el bajo acoplamiento facilita el desarrollo de nuevas funcionalidades. |
| 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* se considera suficientemente segura para este proceso. Las claves deberían 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. |
| 12 | AT012 | El uso de un servicio web REST permitirá el acceso automático al sistema a través de software cliente. |
| 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. |
| 15 | AT015 | El uso de una base de datos SQL Relacional nos ofrece acceso concurrente. |
| 19 | AT019 | El uso de una base de datos relacional proporciona suficiente eficiencia como para aseguras que en cuanto el usuario tenga sus claves de acceso se va a poder identificar en el sistema. |
| 23 | AT023 | La interfaz ChangeInfo no permite cambiar la contraseña a una débil. El cliente web informará al usuario de como debe ser la contraseña. |
| 24 | AT024 | El uso de Hibernate por el componente DBManagement nos asegura que lo que introduzca el usuario será tratado exclusivamente como datos. En ningún caso se formará la consulta con lo introducido por el usuario. |
| 27 | AT027 | El sencillo cliente html ofrece una accesibilidad mínima AA. |
| 28 | AT028 | Se usa un cliente web de lo más sencillo para que la curva de dificultad sea muy baja. |
| 29 | AT029 | El desarrollo de un servicio web REST basado en formatos JSON ofrece interoperabilidad con otros lenguajes o sistemas. |
| 39 | AT039 | Se usará el formato interoperable JSON para las peticiones del Servicio Web Rest. |
| 43 | AT043 | Se usará el framework Spring Boot para su desarrollo. |

## InciManager

### Presentación principal

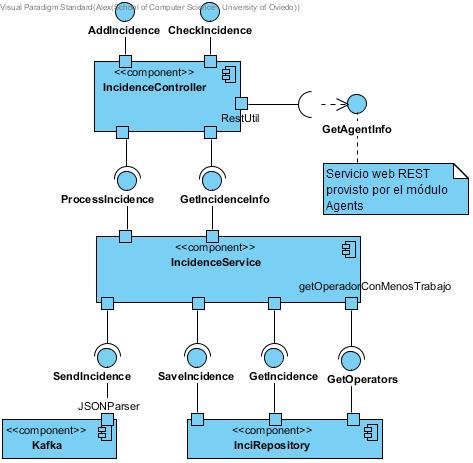


Figura 6. Vista de InciManager

### Catálogo de elementos

#### Elementos

|  |  |
| --- | --- |
| Elemento | Propiedades |
| IncidenceController | Se accede a través de dos servicios web: AddIncidence, que permite notificar al sistema de una nueva incidencia, y CheckIncidence, que permite comprobar el estado de una incidencia. Se implementará un cliente web que consuma los dos servicios. |
| IncideceService | Capa intermedia que procesa y devuelve las incidencias que le solicita IncidenceController. |
| Kafka | Componente que abstrae el uso de Apache Kafka para comunicar este módulo con InciDashboard. |
| InciRepository | Encapsula todas las operaciones de base de datos usando interfaces para permitir el acceso a la base de datos. |

#### Relaciones

Un cliente web consume las operaciones de IncidenceController para ofrecer a los agentes añadir nuevas incidencias al sistema.

IncidenceController permite añadir nuevas incidencias al sistema a través de la interfaz AddIncidence, para ello comprueba que el agente esté registrado comunicándose con el módulo Agents a través de un servicio web REST implementado por este último. También comprueba que los datos sean válidos y con el formato correcto. Si el agente está registrado, puede enviar incidencias y los datos son correctos se comunica con IncidenceService a través de la interfaz ProcessIncidence y le pasa la incidencia creada.

IncidenceService procesa la incidencia y la envía por Apache Kafka a través de la interfaz SendIncidence provista por el componente Kafka. También guarda ciertas incidencias en la base de datos, a través de la interfaz SaveIncidence que provee el componente InciRepository.

IncidenceController también permite consultar las incidencias enviadas por el usuario, después de comprobar que el Agente está registrado en el sistema a través del servicio web GetAgentInfo, pide las incidencias del agente a IncidenceService a través de GetIncidenceInfo y este la recupera de la base de datos a través de la interfaz GetIncidence, provista por el componente InciRepository.

#### Interfaces / Puertos

##### IncidenceController

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Tipo | Tecnología | Propiedades |
| AddIncidence | Interface | Servicio Web | Permite añadir una nueva incidencia través de una petición POST y un objeto JSON. |
| CheckIncidence | Interface | Servicio Web | Devuelve los datos de las incidencias que ha enviado un agente, el agente se identifica con la conbinación: (identificador, contraseña, tipo). |
| GetAgentInfo | Interface (Requerida) | Servicio Web REST | Devuelve la información del agente que coincida con (identificador,contraseña,tipo) o 404 si no existe. |
| ProcessIncidence | Interface (Requerida) | Invocación a método | Interfaz de IncidenceService que permite seguir procesando la incidencia enviada por el agente. |
| GetIncidenceInfo | Interface (Requerida) | Invocación a método | Interfaz de IncidenceService que devuelve la información de las incidencias enviadas por un agente. |
| RestUtil | Port | Invocación a método | Permite realizar peticiones REST a Agents. |

##### IncidenceService

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Tipo | Tecnología | Propiedades |
| ProcessIncidence | Interface | Invocación a método | Procesa la incidencia que le llega. |
| GetIncidenceInfo | Interface | Invocación a método | Devuelve las incidencias de un agente. |
| SendIncidence | Interface (Requerida) | Invocación a método | Envía la incidencia por Apache Kafka. |
| SaveIncidence | Interface (Requerida) | Acceso a BBDD | Guarda en la base de datos la incidencia. |
| GetIncidence | Interface (Requerida) | Acceso a BBDD | Devuelve de la base de datos las incidencias de un agente. |
| getOperador-ConMenosTrabajo | Port | Invocación a método | Devuelve el operador con menos trabajo de la base de datos. |

##### Kafka

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Tipo | Tecnología | Propiedades |
| SendIncidence | Interface | Incoación a método | Usando la tecnología de mensajería Apache Kafka, implementada por la Fundación Apache, envía y registra en el log la incidencia en formato JSON. |
| JSONParse | Port | Invocación a método | Transforma una incidencia a formato JSON. |

##### InciRepository

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Tipo | Tecnología | Propiedades |
| SaveIncidence | Interface | Acceso a BBDD | Guarda en una base de datos SQL Relacional la incidencia pasada. |
| GetIncidence | Interface | Acceso a BBDD | Recupera de la base de datos SQL Relacional la incidencia pedida. |
| GetOperators | Interface | Acceso a BBDD | Recupera de la base de datos SQL Relaciona los operadores guardados. |

#### Comportamiento

##### IncidenceController

Se implementará un cliente web sencillo que consuma las operaciones implementadas por IncidenceController.

IncidenceController implementa un servicio web que nos permite enviar y consultar incidencias.

Para poder enviar incidencias se va a chequear que los agentes están dados de alta en el sistema y tienen permisos para enviar incidencias, el módulo *Agents* proporciona estos datos.

Validará los datos recibidos y rellenará los datos opcionales (Ej: ubicación de un sensor). Si ocurre un error durante la validación de los datos le mostrará un error al usuario. Un error propio, no un error del sistema, de manera que evitemos fugas de información.

Para poder consultar incidencias también se va a chequear que los agentes están dados de alta en el sistema, el módulo *Agents* proporciona estos datos. Y el sistema devolverá únicamente las incidencias del usuario identificado.

##### IncidenceService

Añade a la incidencia la información propia de la lógica de negocio, como el estado de la incidencia, el operario asignado o la caducidad de esta. Envía la incidencia a Kafka y envía algunas incidencias a InciRepository. Enviará a InciRepository las incidencias que puedan ser consultadas luego por el agente que lo envió, por ejemplo, una persona o una entidad.

##### Kafka

Encapsula el comportamiento necesario para poder enviar incidencias a través de Apache Kafka. Estas incidencias serán parcheadas a JSON y enviadas en este formato a través de Kafka al módulo InciDashboard.

##### InciRepository

Encapsula todos los accesos a la base de datos. Este módulo usará la tecnología Hibernate para acceder a la BBDD. El uso de esta tecnología nos permite abstraernos aún más de la BBDD y nos proporciona seguridad extra.

### Diagrama contextual

Ver 9.1.

### Justificación de las decisiones

| Escenario | Atributos de calidad | Justificación |
| --- | --- | --- |
| 17 | AT017 | La simplicidad del diseño y la eficiencia de una base de datos relacional, nos permite asegurar que el usuario va a ser capaz de consultar el estado de su incidencia apenas unos segundos después de que haya sido interceptada por el sistema. |
| 18 | AT018 | El uso de Apache Kafka nos permite que la incidencia le llegue al módulo InciDashboard en menos de 30 segundos. |
| 24 | AT024 | El uso de Hibernate en InciRepository nos asegura que lo que el usuario introduzca será tratado como datos exclusivamente. |
| 27 | AT027 | El cliente HTML que se implemente deberá cumplir, al menos, con el grado de accesibilidad AA. |
| 28 | AT028 | El cliente HTML será muy sencillo, por lo que la curva de aprendizaje será mínima. |
| 30 | AT030 | El uso de Apache Kafka nos permite que, en un futuro, otro desarrollador implemente un sistema que pueda consumir los eventos producidos. |
| 44 | AT044 | Se utilizará la tecnología Apache Kafka para integrar este módulo con InciDashboard. |
| 45 | AT045 | Se utilizará el framework Spring Boot para desarrollar el módulo, debido a su facilidad y disponibilidad de funcionalidad. |
| 46 | AT046 | Este sistema enviará por Kafka en formato JSON. |

## InciDashboard

### Presentación principal

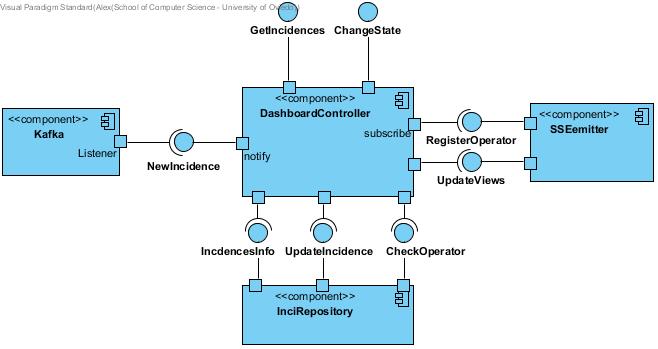


Figura 7. Vista de InciDashboard

### Catálogo de elementos

#### Elementos

|  |  |
| --- | --- |
| Elemento | Propiedades |
| DashboardController | Devuelve las incidencias registradas en el sistema y permite cambiar el estado de estas. Cuando es notificado de una nueva incidencia se comunica con SSEemitter para actualizar las vistas de los operarios. |
| Kafka | Actúa de listener en el bus de Apache Kafka y notifica a DashboardController cuando llega una nueva incidencia. |
| InciRepository | Encapsula todas las operaciones de base de datos usando interfaces para permitir el acceso a la base de datos. |
| SSEemitter | Encapsula el uso de la tecnología SSEemitter de la W3C, la cual nos permite actualizar en tiempo real los clientes web. O sea, las vistas de los operarios. |

#### Relaciones

Junto a este subsistema se implementará un sencillo cliente HTML que consuma los servicios web REST que ofrece DashboardController y pueda ofrecer a los operarios distintas vistas de las incidencias actuales.

Cuando IncidenceManager registre una incidencia, esta será enviada por ApacheKafka. El componente Kafka actuará de listener y cuando le llegue una incidencia esta será notificada a DashboardController a través de su interfaz NewIncidence. Luego actualizará las vistas de los operarios haciendo uso del componente SSEemitter a través de un interfaz UpdateViews.

Cuando un operario acceda al cliente web, este se identificará con sus claves de acceso, a través de la interfaz GetIncidences. DashboardController comprobará que los datos sean correctos haciendo uso de la interfaz CheckOperator. Si los datos son correctos, el sistema devolverá todas las incencias, haciendo uso de la interfaz IncidencesInfo provista por el componente InciRepository. También registrará al nuevo operario en el componente SSEemitter para que, cuando lleguen nuevas incidencias, se le actualice el cliente en tiempo real.

#### Interfaces / Puertos

##### DashboardController

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Tipo | Tecnología | Propiedades |
| GetIncidences | Interface | Servicio web REST | Permite obtener todas las incidencias actuales no resueltas del sistema. |
| ChangeState | Interface | Servicio web REST | Permite cambiar el estado de una incidencia. |
| notify | Port |  | Permite notificar a DashboardController que ha llegado una nueva incidencia. |
| subscribe | Port |  | Puerto que subscribe a los operadores en el componente sseEmitter. |

##### Kafka

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Tipo | Tecnología | Propiedades |
| Listener | Port |  | Listener de Kafka que se activa cuando recibe una incidencia a través de Apache Kafka. |
| NewIncidence | Interface (Requerida) | Invocación a método | Notifica de una nueva incidencia a DashboardController. |

##### InciRepository

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Tipo | Tecnología | Propiedades |
| IncidencesInfo | Interface | Acceso de BBDD | Permite recuperar de la base de datos las incidencias. |
| UpdateIncidence | Interface | Acceso de BBDD | Permite actualizar una incidencia. |
| CheckOperator | Interface | Acceso de BBDD | Permite comprobar si un operador (usuario/contraseña) se encuentra en la base de datos. |

##### SSEemitter

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Tipo | Tecnología | Propiedades |
| RegisterOperator | Interface | Invocación a método | Permite registrar un cliente para ser actualizado más tarde. |
| UpdateViews | Interface | Invocación a método | Permite actualizar las vistas de los clientes. |

#### Comportamiento

##### DashboardController

Se implementará un cliente web sencillo que consuma los servicios Web REST implementados por DashboardController.

DashboardController permite al operario consultar las incidencias actuales y en tiempo real. También permite actualizar el estado de una incidencia. Para conseguir actualizar las vistas de los operarios en tiempo real, DashboardController implementa una interfaz NewIncidence, que le permite al listener comunicar a este componente cuando llega una nueva incidencia.

Haciendo uso del módulo SSEemitter, este componente es capaz de actualizar en tiempo real las vistas de los usuarios.

##### Kafka

Este componente se encargará de realizar el rol de listener del stream Apache Kafka. Cuando reciba una nueva incidencia este se la comunicará a DashboardController para notificarle de que una nueva incidencia ha llegado.

##### InciRepository

Encapsula todos los accesos a la base de datos. Este módulo usará la tecnología Hibernate para acceder a la BBDD. El uso de esta tecnología nos permite abstraernos aún más de la BBDD y nos proporciona seguridad extra.

##### SSEemitter

SSEemitter usará la tecnología estándar del W3C de los Server-Sent Events, los usaremos para poder proporcionar a las vistas eventos enviados por el servidor a los clientes HTML de los usuarios. Con esto lograremos una actualización en tiempo real y gestionada por el servidor.

### Diagrama contextual

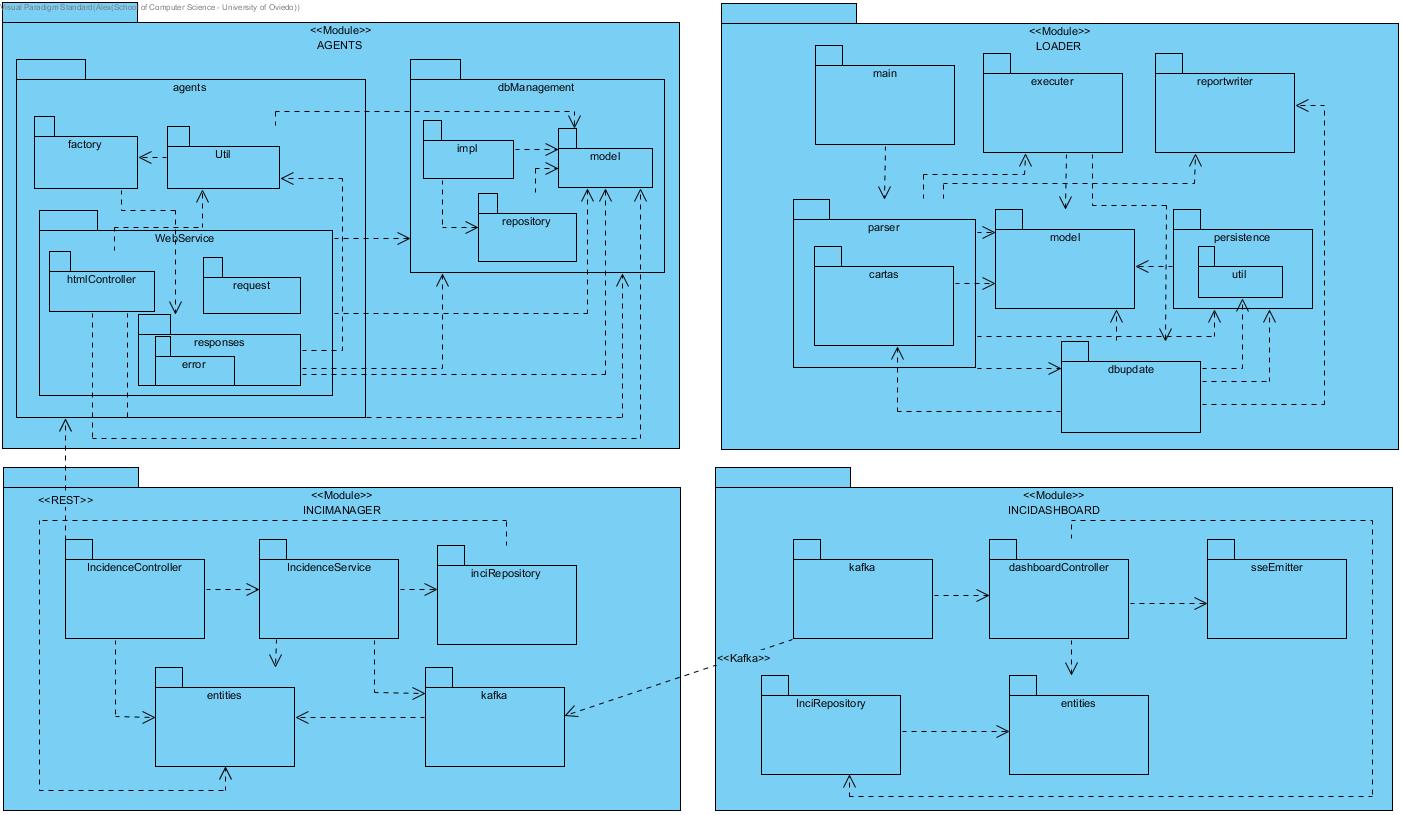
Ver 9.1.

### Justificación de las decisiones

|  |  |  |
| --- | --- | --- |
| Escenario | Atributos de calidad | Justificación |
| 16 | AT016 | Al delegar en el cliente web las vistas podemos modificar y añadir nuevas vistas cambiando simplemente el cliente. Además, el uso de JS nos proporciona una gran versatilidad. |
| 18 | AT018 | Mediante el uso del estándar SSE, su implementación en Spring Boot SSEEmitter y el uso de Apache Kafka se espera poder ofrecer un rendimiento óptimo. |
| 20 | AT020 | El componente DashboardController, con la comunicación directa de InciRepository y la interfaz CheckOperator, evitará cualquier acceso no autorizado al sistema. |
| 21 | AT021 | El sistema comprobará en la base de datos InciDatabase si los datos del operario son correctos. DashboardController hará uso de la interfaz CheckOperator para este fin. |
| 24 | AT024 | El uso de Hibernate en InciRepository nos asegura que lo que el usuario introduzca será tratado como datos exclusivamente. |
| 27 | AT027 | El cliente HTML que se implemente deberá cumplir, al menos, con el grado de accesibilidad AA. Estará implementado en HTML5, CSS3 y JS para mayor compatibilidad. |
| 28 | AT028 | El cliente HTML será muy sencillo, por lo que la curva de aprendizaje será mínima. |
| 31 | AT031 | El uso de Apache Kafka nos permite que, en un futuro, otro desarrollador implemente un sistema que pueda producir los eventos consumidos. |
| 44 | AT044 | Se utilizará la tecnología Apache Kafka para integrar este módulo con InciManager. |
| 45 | AT045 | Se utilizará el framework Spring Boot para desarrollar el módulo, debido a su facilidad y disponibilidad de funcionalidad. |
| 46 | AT046 | Este sistema espera recibir por Kafka un formato JSON. |

## Vista de Paquetes

### Presentación principal



**figura 8. Vista de paquetes.**

### Catálogo de elementos

#### Elementos

##### AGENTS

|  |  |
| --- | --- |
| Elemento | Propiedades |
| agents | Paquete encargado de encapsular la lógica del módulo. |
| factory | Paquete que sigue el estilo de diseño Factory. |
| util | Paquete de utilidades. |
| WebService | Paquete encargado de encapsular el servicio Web Rest que ofrece el módulo. |
| htmlController | Paquete encargado de encapsular los controladores HTML. |
| request | Paquete encargado de encapsular las distintas respuestas que puede devolver la aplicación. |
| responses | Paquete encargado de encapsular las distintas respuestas que puede devolver la aplicación. |
| error | Paquete encargado de encapsular los distintos errores que puede devolver la aplicación. |
| dbManagement | Paquete encargado de encapsular los accesos a BBDD. |
| impl | Paquete encargado de encapsular la implementación de dbManagement. |
| model | Paquete encargado de encapsular el modelo de la aplicación. |
| repository | Paquete encargado de encapsular los accesos a BBDD. |

##### LOADER

|  |  |
| --- | --- |
| Elemento | Propiedades |
| main | Paquete principal desde el que se ejecuta la aplicación. |
| executer | Paquete que sigue el patrón de diseño command. |
| reportwriter | Paquete encargado de encapsular la lógica del logger. |
| parser | Paquete encargado de encapsular el parser de CSV a objetos java. |
| cartas | Paquete encargado de encapsular la lógica de creación de las cartas. |
| model | Paquete encargado de encapsular el modelo de la aplicación. |
| persistence | Paquete encargado de encapsular los accesos a BBDD. |
| útil | Paquete de utilidades. |
| dbupdate | Paquete encargado de encapsular los accesos a BBDD. |

##### INCIMANAGER

|  |  |
| --- | --- |
| Elemento | Propiedades |
| IncidenceController | Paquete encargado de encapsular los servicios que ofrece el módulo. |
| IncidenceService | Paquete encargado de encapsular la lógica de los servicios que ofrece el módulo. |
| InciRepository | Paquete encargado de encapsular los accesos a BBDD. |
| entities | Paquete encargado de encapsular el modelo de la aplicación. |
| kafka | Paquete encargado de encapsular la tecnología Apache Kafka, en este caso un producer. |

##### INCIDASHBOARD

|  |  |
| --- | --- |
| Elemento | Propiedades |
| Kafka | Paquete encargado de encapsular la tecnología Apache Kafka, en este caso un listener. |
| dashboardController | Paquete encargado de encapsular los servicios que ofrece el módulo. |
| sseEmitter | Paquete encargado de encapsular la tecnología SSEEmitter para los Server-Send Events. |
| inciRepository | Paquete encargado de encapsular los accesos a BBDD. |
| entities | Paquete encargado de encapsular el modelo de la aplicación. |

#### Relaciones

##### AGENTS

El paquete agents depende del paquete dbManagement

##### LOADER

El paquete main depende del paquete parser, el cual lee el fichero y lo convierte a objetos java.

Parser, a su vez, depende del paquete executer, reportwriter, model, dbUpdate y persistence. El submodulo de parser, cartas, depende también de model.

##### INCIMANAGER

IncidenceController depende del paquete IncidenceService, del cual se nutre de la lógica que este último implementa y de el modelo encapsulado en entities. También tiene una dependencia externa hacia el paquete agents, del módulo AGENTS, con el cual se comunica mediante el servicio web REST implementado por Agents.

IncidenceService depende de InciRepository, para el acceso a la BBDD y del paquete Kafka para el acceso al stream Kafka. También depende del modelo encapsulado en entitites.

InciRepository y Kafka dependen del modelo encapsulado en entities.

##### INCIDASHBOARD

Kafka depende del paquete Kafka de InciManager para recibir en tiempo real las incidencias. Se comunican a través de Apache Kafka.

dashboardController depende del paquete sseEmitter, para actualizar las vistas de los operarios en tiempo real, de entities, para acceder al modelo y de InciRepository para acceder a la base de datos.

InciRepository depende del modelo encapsulado en entities.

#### Interfaces / Puertos

N/A

#### Comportamiento

##### AGENTS

El paquete Agents se encarga de encapsular la lógica del módulo. El paquete factory sigue el estilo Factory para instanciar objetos del paquete responses.error. Util es un paquete con distintas funciones de utilidad.

WebService es el paquete encargado de gestionar el servicio web rest que ofrece el módulo. HtmlController es un paquete con controladores web que responden en HTML. Request es un paquete que encapsula las distintas peticiones que soporta el servicio web. Responses y responses.error constituyen las respuestas que puede devolver el servicio y las respuestas a peticiones erróneas, respectivamente.

dbManagement encapsula la lógica necesaria para el acceso a BBDD. El paquete impl encapsula la implementación de dbManagement. El paquete model contiene el modelo de la aplicación. Repository es el encargado de encapsular las clases con estereotipo repository de Spring Boot.

##### LOADER

Main, paquete principal que coordina la lógica desde el inicio hasta el final del programa.

Executer, paquete que sigue el patrón de diseño command, provee lógica para guardar los datos. ReportWriter encapsula la lógica del logger. Model encapsula el modelo de datos de la aplicación. DbUpdate se encarga de actualizar la base de datos.

Parser encapsula la lógica para parsear de un fichero a unos objetos java. Cartas encapsula la lógica para crear las cartas para los usuarios.

Persistence se encarga de encapsular los accesos a base de datos. Util es un paquete de utilidades.

##### INCIMANAGER

IncidenceController se encarga de encapsular los controladores que ofrecen los servicios del módulo.

IncidenceService se encarga de guardar la lógica principal del módulo.

InciRepository encapsula los accesos a base de datos.

Entities encapsula el modelo de la aplicación.

Kafka encapsula el acceso a Apache Kafka. Implementa un producer.

##### INCIDASHBOARD

DashboardController se encarga de encapsular los controladores REST que ofrecen los servicios web del módulo.

InciRepository encapsula los accesos a base de datos.

Entities encapsula el modelo de la aplicación.

Kafka encapsula el acceso a Apache Kafka. Implementa un listener.

sseEmitter encapsula la lógica necesaria para hacer uso de la tecnología Server-Send Events.

### Diagrama contextual

Ver 9.1

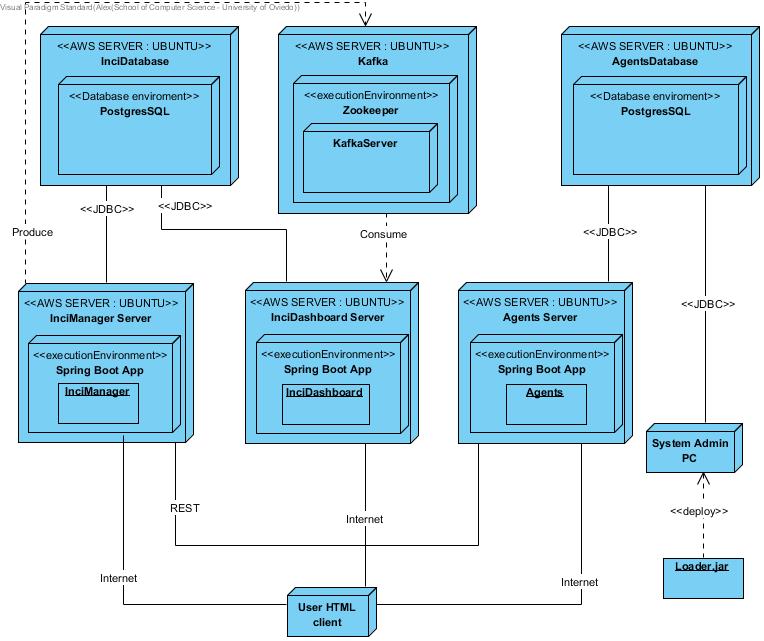
### Justificación de las decisiones

La principal justificación para el diseño de los paquetes en InciManager e InciDashboard es su bajo acoplamiento, la inexistencia de dependencias cíclicas y la simplicidad del diseño, lo que nos permitirá implementarlo y cambiar alguna cosa de manera sencilla. Las decisiones que han llevado a este diseño son:

| Escenario | Atributos de calidad | Justificación |
| --- | --- | --- |
| 4 | AT004 | El formato de salida está encapsulado en el paquete responses. |
| 6 | AT006 | El formato de los web services se encuentra encapsulado en el paquete WebService. |
| 13 | AT013 | El sistema ofrece un diseño simple y con bajo acoplamiento. |

## Vista de Despliegue

### Presentación principal



**Figura 9. Vista de despliegue.**

### Catálogo de elementos

#### Elementos

|  |  |
| --- | --- |
| Tecnologías | Definiciones |
| ASW SERVER: UBUNTU | Máquina virtual de Amazon Web Service, con SO Linux Ubuntu. |
| Zookeeper | Tegnología de apache para facilitar la creación de clusters distribuidos. |
| PostgreSQL | Sistema gestor de BBDD relacional y SQL. |
| Spring Boot App | Framework de java. |
| KafkaServer | Sistema de mensajería creado por Apache. |

|  |  |
| --- | --- |
| Elemento | Propiedades |
| InciDatabase | Servidor en la nube que se encarga de almacenar los datos de Incidencias en una BBDD PostgreSQL. |
| AgentsDatabase | Servidor en la nube que se encarga de almacenar los datos de Agentes en una BBDD PostgreSQL. |
| Kafka | Servidor en la nube que se encarga de albergar el sistema de mensajería Apache Kafka. |
| InciManager Server | Servidor en la nube donde se va a desplegar el módulo InciManager. |
| InciDashboard Server | Servidor en la nube donde se va a desplegar el módulo InciDashboard. |
| Agents Server | Servidor en la nube donde se va a desplegar el módulo Agents. |
| System Admin PC | Ordenador del administrador del sistema que se va a encargar de ejecutar el módulo Loader. |
| User HTML Cliente | Cliente HTML (Firefox, Chrome…) de los usuarios (Operadores, Agentes…), va a ejecutar/visualizar los distintos clientes web. |

#### Relaciones

User HTML client se comunica con InciManager, InciDashboard y Agents mediante protocolos HTTP.

System Admin PC se comunica con AgentsDatabase mediante el protocolo Java Database Connectivity, en adelante JDBC, para guardar los datos de los Agentes.

Agents Server se comunica con AgentsDatabase mediante JDBC para recuperar los datos de los agentes y actualizarlos.

InciDashboard Server se comunica mediante JDBC con InciDatabase para recuperar y actualizar los datos de las Incidencia y los Operadores.

InciManager Server se comunica mediante servicios REST con Agents Server para comprobar los datos de los Agentes. También con InciDatabase mediante JDBC para cargar y recuperar los datos de Incidencia y recuperar los datos de los Operadores. También se comunica con Kafka mediante StreamKafka para mandar las incidencias por el sistema de mensajería Apache Kafka, el cual se comunica con InciDashboard Server mediante StreamKafka.

#### Interfaces / Puertos

N/A

#### Comportamiento

System Admin PC se encargará de ejecutar el programa estilo BASH Loader, cuando el administrador tenga que cargar nuevos datos de Agentes.

User HTML client se encargará de visualizar e interpretar los distintos clientes web que proporcionan InciManager, InciDashboard y Agents.

InciManager Server se encarga de albergar y ejecutar el módulo InciManager, también provee el cliente web de este.

InciDashboard Server se encarga de albergar y ejecutar el módulo InciDashboard, también provee el cliente web de este.

Agents Server se encarga de albergar y ejecutar el módulo Agents, también provee el cliente web de este.

Kafka se encarga de albergar y ejecutar el sistema de mensajería Apache Kafka.

InciDatabase se encarga de albergar y ejecutar el un servidor de base de datos PostgreSQL.

AgentsDatabase se encarga de albergar y ejecutar el un servidor de base de datos PostgreSQL.

### Diagrama contextual

Ver 9.1

### Justificación de las decisiones

Se ha seguido un estilo de despliegue ServerLess. Las decisiones que han llevado a este diseño son:

| Escenario | Atributos de calidad | Justificación |
| --- | --- | --- |
| 14 | AT014 | Aunque el diseño tenga poco acoplamiento no es demasiado complejo, por lo que se puede desplegar el sistema de forma rápida. |
| 30 | AT030 | Al haber albergado el sistema Kafka en un servidor propio, al que se puede acceder desde internet, en un futuro, con la configuración y permisos adecuados, una aplicación de terceros podría subscribirse al canal de mensajería. |
| 31 | AT031 | Al haber albergado el sistema Kafka en un servidor propio, desde el que se puede acceder a internet, en un futuro, con la configuración y permisos adecuados, una aplicación de terceros podría producir mensajes en el canal de mensajería. |
| 32 | AT032 | Este bajo acoplamiento entre máquinas nos permite |
| 33 | AT033 | Este diseño permite el despliegue independiente de módulos, pues el despliegue de un módulo no depende de otro. |
| 34 | AT034 | Al haberlo implementado en Java, podemos migrar el sistema a cualquier SO. |
| 35 | AT035 | Al haberlo despegado en una máquina virtual de AWS podemos asegurar que el sistema será tolerante ante fallos mecánicos (Discos duros rotos, CPUs quemadas, subidas de tensión…) |
| 36 | AT036 | Al haberlo despegado en una máquina virtual de AWS podemos aumentar o disminuir los recursos de esta a nuestras necesidades. |

# 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.