

Escuela de Ingeniería Informática, Univ. Oviedo

12 de febrero de 2018

GestUsers: Sistema de Gestión de Usuarios

**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 GestUsers. Descripción del trabajo práctico (2018)*

Descripción de la práctica primera a realizar por los grupos de trabajo de la asignatura de Arquitectura del Software durante el curso 2017-18.

**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** |

**GestUsers: Sistema de Gestión de Usuarios**

**Autores**: **D. Aquilino Adolfo Juan Fuente**

Doctor Ingeniero en Informática

Profesor del Departamento de Informática de la Universidad de Oviedo

**D. Jose Emilio Labra Gayo**

Doctor Ingeniero en Informática

Profesor del Departamento de Informática de la Universidad de Oviedo

**Juan Luis Mateo Cerdán**

PhD. Computer Science

Dept. Computer Science, University of Oviedo

**Herminio García González**

Dept. Computer Science, University of Oviedo

**Fecha**: 12 de febrero de 2018

**Versión**:

Tabla de contenido

[1 Introducción y Objetivos 5](#_Toc506219137)

[2 Requisitos 6](#_Toc506219138)

[2.1 Loader 6](#_Toc506219139)

[2.2 Agents 7](#_Toc506219140)

[3 Metodología usada 8](#_Toc506219141)

[4 Identificación de *Stakeholders* 9](#_Toc506219142)

[4.1 Alumnos que realizan la práctica 9](#_Toc506219143)

[4.2 Administrador del Sistema 9](#_Toc506219144)

[4.3 Agentes 9](#_Toc506219145)

[4.4 Responsables políticos del portal 10](#_Toc506219146)

[4.5 Profesores de la asignatura 10](#_Toc506219147)

[5 Atributos de calidad 11](#_Toc506219148)

[5.1 Lista de atributos de calidad 11](#_Toc506219149)

[5.2 Atributos de calidad e Interesados 12](#_Toc506219150)

[6 Restricciones 14](#_Toc506219151)

[6.1 Restricciones técnicas 14](#_Toc506219152)

[6.2 Restricciones organizativas 14](#_Toc506219153)

[7 Ámbito del sistema y contexto 15](#_Toc506219154)

[8 Escenarios de calidad 17](#_Toc506219155)

[9 Vistas 20](#_Toc506219156)

[9.1 Contexto 20](#_Toc506219157)

[9.1.1 Presentación principal 20](#_Toc506219158)

[9.1.2 Catálogo de elementos 20](#_Toc506219159)

[9.2 Loader 22](#_Toc506219160)

[9.2.1 Presentación principal 22](#_Toc506219161)

[9.2.2 Catálogo de elementos 22](#_Toc506219162)

[9.2.3 Diagrama contextual 24](#_Toc506219163)

[9.2.4 Justificación de las decisiones 24](#_Toc506219164)

[9.3 Agents 25](#_Toc506219165)

[9.3.1 Presentación principal 25](#_Toc506219166)

[9.3.2 Catálogo de elementos 25](#_Toc506219167)

[9.3.3 Diagrama contextual 27](#_Toc506219168)

[9.3.4 Justificación de las decisiones 27](#_Toc506219169)

[9.4 Vista de Paquetes 28](#_Toc506219170)

[9.4.1 Presentación principal 28](#_Toc506219171)

[9.4.2 Catálogo de elementos 28](#_Toc506219172)

[9.4.3 Diagrama contextual 28](#_Toc506219173)

[9.4.4 Justificación de las decisiones 28](#_Toc506219174)

[9.5 Vista de Despliegue 29](#_Toc506219175)

[9.5.1 Presentación principal 29](#_Toc506219176)

[9.5.2 Catálogo de elementos 29](#_Toc506219177)

[9.5.3 Diagrama contextual 29](#_Toc506219178)

[9.5.4 Justificación de las decisiones 29](#_Toc506219179)

[10 Bibliografía 30](#_Toc506219180)

# Introducción y Objetivos

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

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

El sistema se ha descompuesto en dos partes: Loader para cargar datos de los agentes que pueden enviar incidencias al sistema, y Agents, para consultar y obtener información de los agentes que participan en el sistema. Los estudiantes deberán implementar el software descrito en este documento en dos equipos de 3 ó 4 personas durante 3 semanas. Un equipo desarrollará el módulo Loader y otro equipo desarrollará el módulo Agents.

En los próximos entregables, los estudiantes crearán la arquitectura e implementarán un prototipo del resto del sistema informático de análisis de incidencias del cual formará parte el sistema actual.

# Requisitos

La gestión de usuarios se hará siguiendo un esquema de dos partes:

* Loader: carga los datos de los agentes que podrán enviar incidencias al sistema.
* Agents: consiste en un servicio Web REST que permitirá consultar y obtener información de los agentes que participan en el sistema.

## Loader

Hay diferentes tipos de agentes: personas físicas, entidades, sensores, etc. Cada tipo de usuario estará identificado por una palabra clave. Ejemplos de identificadores del tipo de agente son: "Ciudadano", "Entidad", "Sensor", etc.

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

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

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

1,Person

2,Entity

3,Sensor

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

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

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

## Agents

El servicio Web está pensado para ser utilizado por agentes que puedan conectarse al sistema, por lo que la entrada serán ficheros en formato JSON y las respuestas también serán en formato JSON.

Este sistema podrá disponer de un subsistema de acceso a través de Web para actualizar la clave de cada usuario. Opcionalmente, podrán actualizarse otros campos de los usuarios.

Los agentes deben poder acceder al sistema para logearse y ver su información. 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 agente en el sistema en caso correcto o informe del error en case incorrecto. Tanto los parámetros como la respuesta se enviarán en formato JSON.

Este sistema podrá disponer de un subsistema de acceso a través de Web para actualizar la clave de cada usuario. Opcionalmente, podrán actualizarse otros campos de los usuarios.

El formato de las invocaciones al sistema es el siguiente (obsérvese que se han modificado 2 campos respecto al año pasado):

{

"login": usuario,

"password": password,

"kind": tipo de agente

}

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

{

"name": Nombre,

"location": Coordenadas (opcional),

"email": Email,

"id": identificador,

"kind": tipo de usuario,

"kindCode": código numérico del tipo de usuario,

}

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

# 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.

# Identificación de *Stakeholders*

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

1. Alumnos que realizan la práctica
2. Administrador del Sistema
3. Agentes
4. Responsables políticos del portal
5. Profesores de la asignatura

Así pues, la lista de *stakeholders* queda:

| Código | Stakeholder | Intereses (Módulos) |
| --- | --- | --- |
| ST-01 | Alumnos que realizan la práctica | Ambos |
| ST-02 | Administrador del Sistema | Carga de ficheros |
| ST-03 | Agentes | Comprobaciones de los datos |
| ST-04 | Autoridad de tráfico | Comprobaciones de los datos |
| ST-05 | Profesores de la asignatura | Ambos |

Tabla . Lista de Stakeholders e intereses

Posteriormente se pasa a describir en más detalle cada uno.

## Alumnos que realizan la práctica

Se trata de los equipos de desarrollo.

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.

## Administrador del Sistema

Es la persona que carga los ficheros de datos.

Entre sus objetivos están:

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

## Agentes

Son los usuarios finales del sistema.

Entre sus objetivos están:

* Sencillez de acceso a los datos.
* Ser capaz de acceder de una forma segura.
* Ser capaz de consultar el estado del sistema
* Ser capaz de cambiar su clave
* Ser capaz de cambiar otros datos del usuario (**opcional**)

## Autoridad de tráfico

Está formado por el equipo que se encarga de explotar los resultados de la participación de agentes 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.

# Atributos de calidad

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

* **Disponibilidad**
  + Disponibilidad del sistema para consultar datos 24x7.
* **Modificabilidad**
  + Facilidad de cambio de diversas partes de la aplicación: Cambiar el *parser* de entrada de listas de agentes 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*,… (actualmente todos opcionales excepto la contraseña)
  + Facilidad para modificar partes de la aplicación: otros formatos, mediante negociación de contenido
* **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
* **Seguridad**
  + Garantizar la confidencialidad de los datos.
* **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
* **Usabilidad**
  + El sistema de carga de datos debe poder ser usado por usuarios administradores de sistema familiarizados con herramientas tipo Unix.
* **Interoperabilidad**
  + El presente sistema será usado por el Sistema de Gestión de Usuarios, el cual delegará en el sistema actual la gestión de usuarios. El subsistema *Agents* deberá ser utilizado por un proceso automático para consultar el estado de los usuarios que participan.
* **Simplicidad**
  + Los dos sub-sistemas deberán ser simples y fáciles de desarrollar
* **Desplegabilidad**
  + El sistema debe ser fácilmente desplegable, especialmente en un servidor en la nube.

## Lista de atributos de calidad

| **Código** | **Descripción** | **Tipo de Atributo** | **Módulo afectado** |
| --- | --- | --- | --- |
| **AT001** | Disponibilidad del sistema para consultar datos 24x7 | Disponibilidad | Agents |
| **AT002** | Facilidad de cambio de diversas partes de la aplicación: Cambiar el parser de entrada de listas de Agents. | Modificabilidad | Loader |
| **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 y Agents |
| **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 y Agents |
| **AT014** | El sistema debe ser fácilmente desplegable | Desplegabilidad | Loader y Agents |

Tabla . 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** |
| --- | --- | --- | --- | --- | --- |
| **AT001** | X |  | X | X | X |
| **AT002** | X | X |  |  | X |
| **AT003** | X | X |  |  | X |
| **AT004** | X | X |  |  | X |
| **AT005** | X |  | X |  | X |
| **AT006** | X |  | X | X | X |
| **AT007** | X | X |  |  | X |
| **AT008** | X | X |  |  | X |
| **AT009** | X | X |  |  | X |
| **AT010** | X |  | X | X | X |
| **AT011** | X | X |  |  | X |
| **AT012** | X |  |  | X | X |
| **AT013** | X |  |  | X | X |
| **AT014** | X | X |  |  | X |

Tabla . 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 (ST001) tiene conocimientos de Java |
| **TC002** | Se utilizará una base de datos relacional para almacenar los datos | Se asume que el equipo de desarrollo (ST001) 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. |
| **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 texto plano | Con el fin de facilitar la implementación se propone generar cartas personalizadas mediante texto plano. El equipo de desarrollo puede opcionalmente implementar otros formatos |
| **TC007** | 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. |
| **TC008** | 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. |

Tabla . Restricciones técnicas

## Restricciones organizativas

| **Código** | **Restricción** | **Motivación** |
| --- | --- | --- |
| **OC001** | Cada sub-sistema será implementado por un equipo pequeño de estudiantes. | El tamaño de los equipos es de 5 y 4 estudiantes con el fin de que los estudiantes puedan aprender a desarrollar software de forma colaborativa mediante un proyecto simple. |
| **OC002** | La estructura de la base de datos será la misma para los 2 sub-sistemas | 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 |

Tabla . Restricciones organizativas

# Ámbito del sistema y contexto

Para describir la solución se utilizarán diagramas contextuales y texto.

La aplicación está partida en dos procesos:

* Loader: Se encarga de la carga de los ficheros. Utiliza el estilo Batch
* Agents: Se encarga de las comprobaciones de los usuarios. Utiliza el estilo micro-servicios.

Ambos se integran usando el arquitectónico de datos compartidos.

En el diagrama de contexto de la Figura 1, se muestran las principales interfaces de cada sistema. El subsistema DataBase es común a ambos grupos, por tanto, hay que acordar la tecnología, el modelo de base de datos y el modo de acceso.

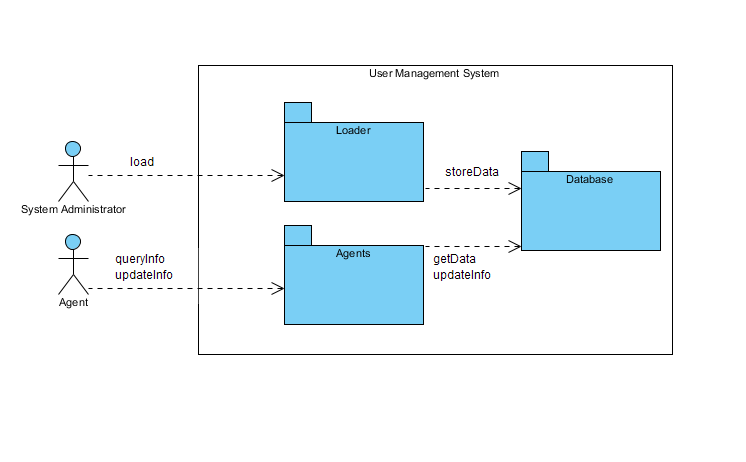


Figura . Contexto de negocio del sistema

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

A destacar:

* Hay una base de datos común.
* Los datos intercambiados usan formato JSON.
* Los procesos de Loader y de Agents son asíncronos.

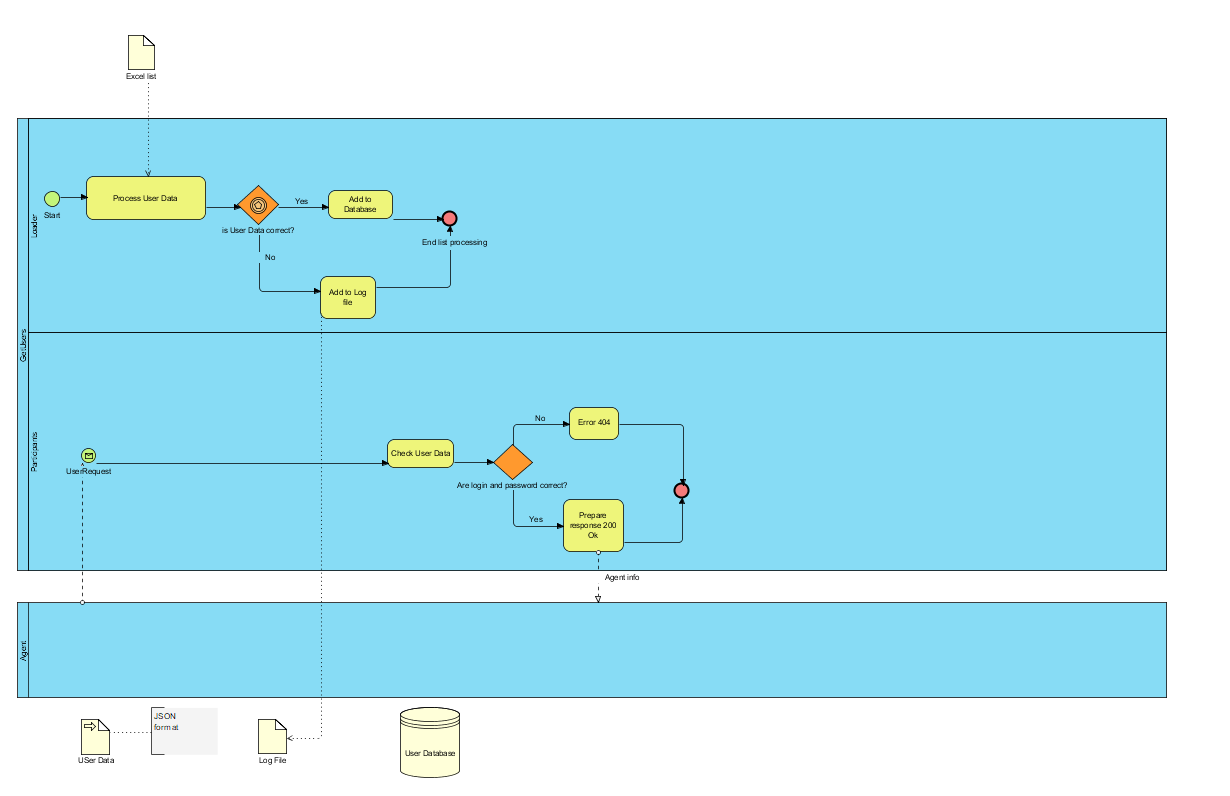


Figura . Diagrama BPMN

# 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 | Sistema de consulta sobre agentes | Realiza una petición de información sobre sus datos | Tiempo de ejecución | Agents | El sistema 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 agentes | 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 agentes | **AT007** |
| 8 | Desarrollador | Cargar un sistema en el sistema (DB) | Desarrollo/  Tiempo de ejecución | Parser, DBUpdate and ReportWriter (Optional) | La carga debe hacerse de manera segura | No es posible acceder a los datos personales de los usuarios salvo el administrador del sistema, que tampoco puede acceder a las contraseñas. | **AT008** |
| 9 | Agentes | Accede a la aplicación | Tiempo de ejecución | Agents | Cada agente puede acceder a sus datos, pero no a los datos de otros | 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 agente 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 Gestión de Agentes | Accede al servicio Web | Tiempo de ejecución | Agents | El sistema 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** |
|  |  |  |  |  |  |  |  |

Tabla . Lista de escenarios de calidad

# Vistas

En los próximos párrafos se describirán algunas de las vistas identificadas y se documentarán de acuerdo a 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 | AT011, AT013 | 11, 13 |
| Loader | ST-01, ST-02, ST-04, ST-05 | AT002, AT003, AT004, AT007, AT008 y AT010, AT011, AT013 | 2, 3, 4, 7, 8, 10, 11, 13 |
| Agents | ST-01, ST-03, ST-04, ST-05 | AT001, AT005, AT006, AT008, AT009, AT012, AT013 | 1, 5, 6, 8, 9, 12, 13 |

En los catálogos y en las vistas se han definido tanto los elementos obligatorios como opcionales (voluntarios). Es responsabilidad de cada equipo eliminar los que no vaya a implementar.

## Contexto

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

### Presentación principal

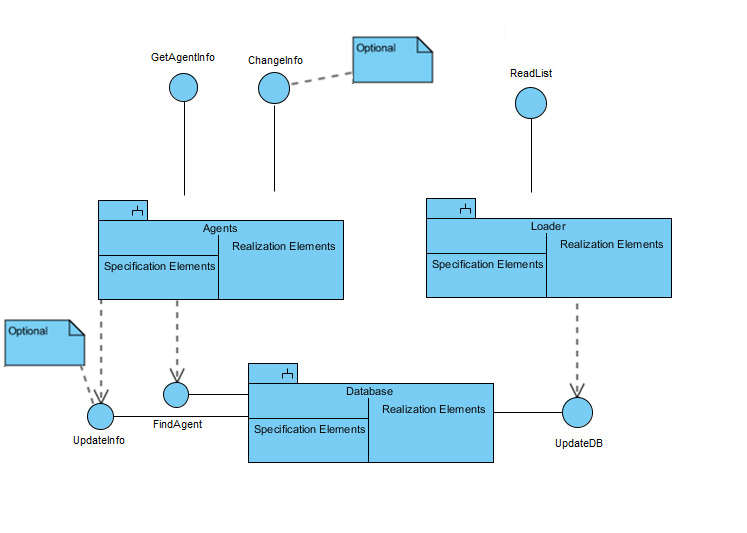


Figura . Context view

### Catálogo de elementos

#### Elementos

|  |  |
| --- | --- |
| Elemento | Propiedades |
| Loader | Se encarga de la introducción de las listas de agentes en el sistema. Lee un fichero con los datos de los agentes. 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. |
| DataBase | Este módulo encapsula los accesos a la base de datos. |

#### 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 *UpdateDB* del módulo *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 *DataBase* a través de la interfaz *FindAgent*.

Opcionalmente se puede implementar la interface *ChangeInfo* que 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*.

#### 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 |
| GetAgentInfo | Interface | Servicio Web | Este servicio se invocará a través de una petición HTTP |

##### DataBase

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Tipo | Tecnología | Propiedades |
| FindAgent | Interface | Invocación a Método | Devuelve los datos de un agente. |
| Insert | Interface | Invocación a Método | Inserta en la base de datos los datos de un agente, incluida su contraseña |
| UpdateInfo | Interface | Invocación a Método | Actualiza la clave del usuario en la base de datos. |

#### Comportamiento

##### Loader

Ver el párrafo 9.2.2.3.4.

Además, puede hacer las siguientes opciones:

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

##### DataBase

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

## Loader

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

### Presentación principal

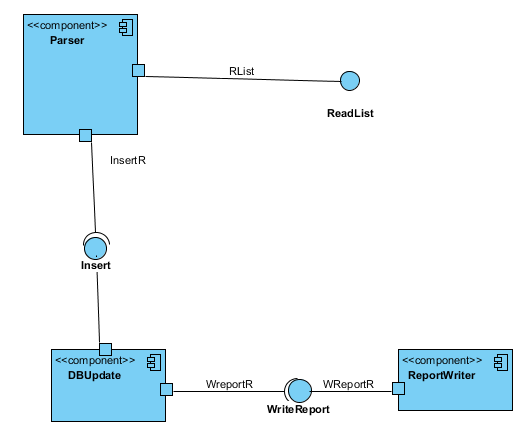


Figura . Vista Loader

### 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 *usuario/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 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*.

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

#### 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. |
| ReaderSingleton | Port |  | Se encarga de llamar a la carga de ficheros y gestionar el caso de que no exista el fichero. |
| 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. |

##### ReportWriter

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Tipo | Tecnología | Propiedades |
| WriteReport | Interface | Invocación a Métodos | Recibe los datos para escribir en el fichero de *log*. |
| WriterReportImpl | Port |  | Implementa la interfaz WriteReport para escribir en el fichero de *log*. |
| ReportWriter | Port |  | Contiene una instancia de WriteReport y de si mismo. |

##### Parser

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

* Nombre (String)
* Email (String con un formato acorde a las convenciones de correo electrónico)
* Tipo (String)
* Identificador (String)
* Localización (String)

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

Este componente también creará los emails personales comunicando al usuario que ha sido añadido al Sistema, 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. |
| 5 | AT005 | La utilización de una base de datos relacional ofrecerá un acceso eficiente a la información de los usuarios |
| 6 | AT006 | 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 de manera individualizada evita que la información pueda ser leída por otros agentes. |
| 8 | AT008 | La utilización de una base de datos relacional con acceso mediante SQL puede permitir a los alumnos verificar que los datos han sido cargados adecuadamente |
| 10 | AT010 | 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 |
| 14 | AT014 | Una aplicación *batch* independiente puede ser ejecutada directamente sin ninguna necesidad especial para su despliegue |

## Agents

### Presentación principal

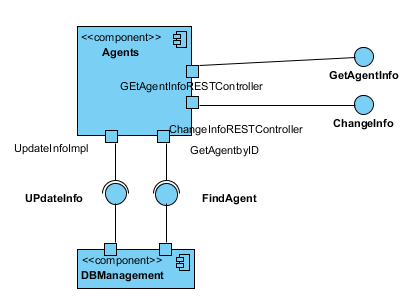


Figura . Vista de Agents

### Catálogo de elementos

#### Elementos

|  |  |
| --- | --- |
| Elemento | Propiedades |
| Agents | Se accede a través de dos servicios web: *GetAgentInfo*, 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 invoca *Agents* utilizando una llamada a un servicio web que es procesada por *GetAgentInfo* (enviado *email/contraseña*) y éste accede a los datos encapsulados en DBManagement mediante la interface *FindAgent*. Si la combinación *email/contraseña* es correcta se devuelve la información del usuario en formato JSON.

(**Opcional**) El usuario (participante) 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*.

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

#### 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*. |
| Check | 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 agente mediante una combinación de datos: *email/contraseña/nuevaContraseña*. |
| Check | 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 |  | -- |
| GetAgentInfo | 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 |  | -- |
| FindAgent | 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": usuario,

"password": password,

"kind": tipo de agente

}

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

{

"name": Nombre,

"location": Coordenadas (opcional),

"email": Email,

"id": identificador,

"kind": tipo de usuario,

"kindCode": código numérico del tipo de usuario,

}

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

(**Opcional**) Se puede implementar un interfaz HTML para que el servicio Web pueda también ser utilizado por personas a través de un navegador Web convencional.

(**Opcional**) El servicio Web puede ser extendido para permitir 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 | AT005 | 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. |
| 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. |

## Vista de Paquetes

### Presentación principal

### Catálogo de elementos

#### Elementos

#### Relaciones

#### Interfaces / Puertos

#### Comportamiento

### Diagrama contextual

Ver …

### Justificación de las decisiones

Las decisiones que han llevado a este diseño son:

| Escenario | Atributos de calidad | Justificación |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## Vista de Despliegue

### Presentación principal

### Catálogo de elementos

#### Elementos

#### Relaciones

#### Interfaces / Puertos

#### Comportamiento

### Diagrama contextual

Ver …

### Justificación de las decisiones

Las decisiones que han llevado a este diseño son:

| Escenario | Atributos de calidad | Justificación |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# 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.