Grupo e1a (Grupo 1 Laboratiorio)

13 de febrero de 2018

Descripción del trabajo práctico (2018)

Realización de la carga de usuarios tal cual indica el enunciado de la práctica 1 del curso 2017-18.

Arquitectura de software

Carga de Usuarios (Loader\_e1a)

**Autores (2017/2018)**

* Ángel Puerta Díaz (UO246998)
* Sergio Domínguez Cabrero (UO250788)
* César Álvarez Rubio (UO251069)
* Alvaro Alonso Quijada (UO251322)

**Autores (2016/2017)**

* Daniel Alba Muñiz (UO245188)
* José Luis Bugallo González (UO244702)
* Ignacio Escribano Burgos (UO227766)
* Daniel Duque Barrientos (UO245553)
* Rubén de la Varga Cabero (UO246977)

**Fecha:**

* 13 Febrero 2018

**Tabla de contenido**

# Introducción y Objetivos

# Con este documento se pretende explicar la estructura y gestión del módulo de carga de usuarios al sistema, mediante el empleo de una base de datos.

# La arquitectura determinada en este documento pertenece a la primera parte de la asignatura de Arquitectura del Software, impartida en el grado de Ingeniería Informática del Software, Escuela de Ingeniería de Informática, Universidad de Oviedo por los autores del documento.

# La totalidad del sistema está compuesta por dos partes: Loader, el encargado de realizar la carga de datos y el cual detallaremos con mayor profundidad en este documento; y Participants, que permite a un ciudadano darse de alta como participante y revisar si puede o no participar.

# Requisitos

Actualmente para realizar la totalidad del proyecto, como ya indicamos anteriormente, es necesario el empleo de dos módulos:

**Loader:**

Este módulo se encarga de cargar los datos de los agentes que podrán enviar incidencias al sistema. A diferencia del curso pasado, hay diferentes tipos de agentes: personas físicas, entidades, sensores, etc. Cada tipo de usuario estará identificado por una palabra clave. Ejemplos de identificadores del tipo de agente son: "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. A diferencia del año pasado, el usuario del sistema será el campo identificador (el año pasado era el correo electrónico).

**Agents:**

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

Al igual que el año pasado, este sistema podrá disponer de un subsistema de acceso a través de Web para actualizar la clave de cada usuario. Opcionalmente, podrán actualizarse otros campos de los usuarios.

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

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

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

*{ “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).

# Identificación de *Stakeholders*

# Existen un total de 5 stakeholders, los cuales son:

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

Cada stakeholder consta de un código y de intereses, éstos últimos son los módulos descritos anteriormente. Así podemos construir una tabla como la siguiente:

|  |  |  |
| --- | --- | --- |
| ***Código*** | ***Stakeholder*** | ***Intereses (Módulos)*** |
| ST-01 | Alumnos que realizan la práctica | Ambos |
| ST-02 | Administrador del sistema | Carga de ficheros |
| ST-03 | Ciudadanos | Comprobación de datos |
| ST-04 | Responsables políticos del portal | Comprobación de datos |
| ST-05 | Profesores de la asignatura | Ambos |

A continuación, describiremos cada uno de ellos.

**Alumnos que realizan la práctica**

Son los equipos de desarrollo del conjunto de módulos, los cuales constan con una serie de objetivos:

-Emplear tecnologías y metodologías conocidas, minimizando así los riesgos utilizando nuevas.

-Aprender técnicas de desarrollo de software de forma colaborativa y profesional.

-Uso de tecnologías similares a las del grupo, de forma que no existan incompatibilidades en el proyecto.

**Administrador del Sistema**

Es la persona la cual carga los ficheros de datos. Posee una serie de objetivos:

-Tecnologías sencillas de ficheros de entrada.

-Ficheros que puedan leerse por personas.

-Poder automatizar el proceso de carga de los datos.

-Poder depurar el proceso de carga si se producen errores.

**Ciudadanos**

Son los usuarios finales del sistema, constan de una serie de objetivos:

-Sencillez de acceso a los datos.

-Poder acceder, de forma segura, desde su hogar.

-Poder consultar el estado del sistema.

-Poder cambiar su información en el sistema como, por ejemplo, la clave (**Opcional**)

**Responsables políticos del portal**

Es el equipo encargado de explotar los resultados de participación en el portal, constan de una serie de objetivos:

-Disponer de información de lo que sucede en el portal cuanto antes.

-Usar tecnologías sencillas e interoperables con otros sistemas.

**Profesores de la asignatura**

Se encargan de los resultados de la práctica, constan de una serie de objetivos:

-Proponen tecnologías para ayudar a los estudiantes a obtener las habilidades relacionadas con la arquitectura del software mediante el desarrollo de un proyecto práctico.

-Introducen a los estudiantes en el desarrollo de software de forma colaborativa y profesional, mediante el desarrollo basado en pruebas o TDD.

-Proponen un trabajo de desarrollo a partir de una documentación, la cual puede ser realizada en el tiempo asignado por los estudiantes de la asignatura.

-Muestran un ejemplo de documentación de arquitectura a los estudiantes.

**Atributos de calidad**

Existen una serie de atributos de calidad identificados para este sistema:

* **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 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

* **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 Participación Ciudadana, el cual delegará en el sistema actual la gestión de usuarios. El subsistema *Participants* deberá ser utilizado por un proceso automático para consultar el estado de los usuarios que quieran participar.

* **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** | **Modulo Afectado** |
| AT001 | Disponibilidad del sistema para consultar datos 24x7 | Disponibilidad | Participants |
| AT002 | Facilidad de cambio de diversas partes de la aplicación: Cambiar el parser de entrada de listas de ciudadanos. | Modificabilidad | CitizensLoader |
| AT003 | Facilidad de cambio de diversas partes de la aplicación: Añadir nueva información al fichero de *log* | Modificabilidad | CitizensLoader |
| AT004 | Facilidad para modificar partes de la aplicación:  Añadir otros formatos de salida o de entrada | Modificabilidad | CitizensLoader |
| 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 | Participants |
| AT006 | Facilidad para cambiar partes de la aplicación: procesar y devolver información en otros formatos mediante negociación de contenido. | Modificabilidad | Participants |
| AT007 | El rendimiento del proceso de carga de datos de los ficheros es razonable (no demasiado lento, pero tampoco crítico) | Rendimiento | CitizensLoader |
| AT008 | El sistema debe garantizar la confidencialidad de los datos de los usuarios | Seguridad | CitizensLoader y Participants |
| AT009 | Debe ser posible chequear que el servicio web se comporta adecuadamente | Testabilidad | Participants |
| AT010 | Debe ser posible chequear el comportamiento del sistema de carga de datos | Testabilidad | CitizensLoader |
| AT011 | El sistema debe poder ser usado por administradores de sistemas familiarizados con herramientas tipo Unix | Usabilidad | CitizensLoader |
| AT012 | El servicio Web debe poder ser utilizado por procesos automáticos que consulten el estado de un usuario | Interoperabilidad | Participants |
| AT013 | El sistema debe ser sencillo y fácil de implementar | Simplicidad | CitizensLoader y Participants |
| AT014 | El sistema debe ser fácilmente desplegable | Desplegabilidad | CitizensLoader y Participants |

**Atributos de calidad e Interesados**

Los atributos de calidad son de interés para los stakeholders, mediante la siguiente tabla se pueden observar estos intereses:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Atributos** | **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 |

**Restricciones**

Existen una serie de restricciones en nuestra aplicación para poder crearla.

**Restricciones técnicas**

|  |  |  |
| --- | --- | --- |
| **Código** | **Restricción** | **Motivación** |
| **TC001** | Programación en lenguaje Java | Se asume que el equipo de desarrollo (ST001) tiene conocimientos de Java |
| **TC002** | Uso de base de datos relacional para almacenar 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** | Servicio web basado en estilo REST con formato de entrada JSON | El estilo REST es fácil de implementar y consumir. |
| **TC004** | Datos de entrada en formato Excel | Excel es un formato de datos bastante popular y existen varias librerías Java para procesar ficheros Excel |
| **TC005** | Formato de salida de los emails personalizados en 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** | Servicio web implementado 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. |

**Restricciones organizativas**

|  |  |  |
| --- | --- | --- |
| **Código** | **Restricción** | **Motivación** |
| **OC001** | Cada sub-sistema es implementado por un equipo pequeño de estudiantes | El tamaño de los equipos será de unos 3 ó 4 estudiantes con el fin de que los estudiantes puedan aprender a desarrollar software de forma colaborativa mediante un proyecto simple. |
| **OC002** | La estructura de la base de datos será la misma para los 2 sub-sistemas | 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 |

**Ámbito del sistema y contexto**

Para describir la solución se emplearán diagramas contextuales y texto.

La aplicación se divide en dos procesos:

-Loader: es la carga de ficheros, usando el estilo Batch.

-Agents: comprueba los datos empleando el estilo micro-servicios.

Ambos se integran usando el arquitectónico de datos compartidos.