



**Govern  
de les Illes Balears**

Vicepresidència i Conselleria  
d'Innovació, Recerca i Turisme  
Direcció General de Desenvolupament  
Tecnològic

una manera de hacer  
**europa** 

Fondo Europeo de  
Desarrollo Regional



Unió Europea

# Diseño Técnico SISTRA2

Diciembre 2017

**Servicios de Administración Electrónica en el Govern de les Illes Balears**

Lot 2 (Servicios electrónicos para la ciudadanía)

**Oficina Técnica de Dirección de Proyecto**

## Control de versiones del documento

Control de Cambios			
Data	Autor	Versión	Cambios
21/12/2017	Indra	v1.0	Diseño Técnico Sistra2 v0
08/01/2018	Indra	V1.1	Errata capa negocio SISTRAHELP
12/02/2018	Indra	V1.2	Integración plugins externos
19/02/2018	Indra	V1.3	Resolución mínima pantallas
19/06/2018	Indra	V1.4	Incorporación de API REST de SISTRAGES e información relativa a roles y permisos

Revisado por		
Nombre	Data	Área, departamento o empresa

Aprobado por		
Nombre	Data	Área, departamento o empresa

Lista de distribución		
Nombre	Área, departamento o empresa	Correo electrónico

## Índex

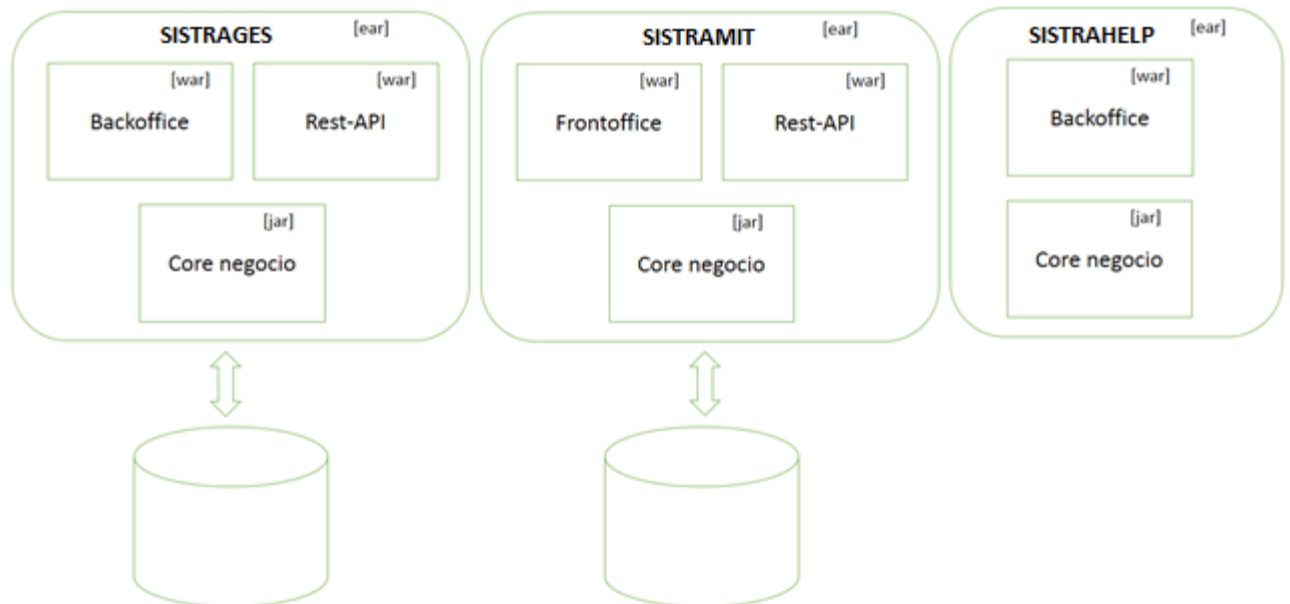
<b>Control de versiones del documento .....</b>	<b>2</b>
<b>1. Introducción .....</b>	<b>4</b>
<b>2. Arquitectura tecnológica.....</b>	<b>5</b>
<b>3. Integración con plugins externos.....</b>	<b>7</b>
<b>4. Esquemas de BBDD.....</b>	<b>9</b>
<b>5. Arquetipo de la aplicación .....</b>	<b>10</b>
<b>6. APIs integración .....</b>	<b>12</b>
6.1. API SISTRAGES.....	12
6.2. API SISTRAMIT .....	13
<b>7. Roles .....</b>	<b>14</b>
<b>8. Infraestructura de soporte .....</b>	<b>15</b>

## 1. Introducción

El objetivo de este documento es definir el diseño técnico de la aplicación de SISTRA2, indicando cuál es su arquitectura tecnológica y su infraestructura de soporte.

## 2. Arquitectura tecnológica

En la arquitectura de la aplicación distinguen 3 módulos diferenciados:



- Gestor de Trámite (SISTRAGES): implementa un backoffice con la consola de administración y de desarrollo de trámites.
- Asistente de Tramitación (SISTRAMIT): implementa un frontoffice con el asistente de tramitación y gestiona la persistencia de los trámites electrónicos en curso hasta que se registren. Se comunica con el SISTRAGES a través de su API para obtener la configuración del sistema y la lógica de los trámites.
- Helpdesk (SISTRAHELP): implementa un backoffice con el Helpdesk que permita resolver las incidencias de tramitación de los ciudadanos.

Por tanto, se genera un EAR para cada módulo: SISTRAGES, SISTRAMIT y SISTRAHELP. De esta forma dependiendo de la carga de trabajo que pueda tener cada módulo, se puede balancear la carga para tener alta disponibilidad. Se prevé que no todos los módulos van a tener la misma carga de trabajo, sobre todo en producción, donde el SISTRAMIT va a tener con diferencia la mayor carga de trabajo, con lo que puede realizar un escalado añadiendo más instancias. Además, la separación en varios EARs permitirá realizar despliegues de los módulos de forma separada.

Las tecnologías usadas en cada módulo son las siguientes:

SISTRAGES	Backoffice	JSF2.2 / PrimeFaces 6.1 Spring 4.3
	Rest-API	Jersey 2.6 Spring 4.3
	Negocio	EJB3 Spring 4.3 JPA2
SISTRAMIT	Frontoffice	Spring MVC 4.3 JSP JS (Zepto, Modernizr, Maskedinput, Haschange, RequireJS, Velocity, MarkUp,...)
	Rest-API	Jersey 2.6 Spring 4.3
	Negocio	EJB3 Spring 4.3 JPA2
SISTRAHELP	Backoffice	JSF2.2 / PrimeFaces 6.1 Spring 4.3
	Negocio	EJB3 Spring 4.3

Respecto a la resolución mínima de las pantallas de las aplicaciones de backoffice se establece el mínimo de 1280x1024 al considerarse una aplicación de escritorio.

### 3. Integración con plugins externos

La integración con plugins externos se realizará mediante APIs basadas en servicios, preferiblemente REST, de forma que no exista acoplamiento entre SISTRA2 y los plugins externos o componentes que ofrecen dichos servicios.

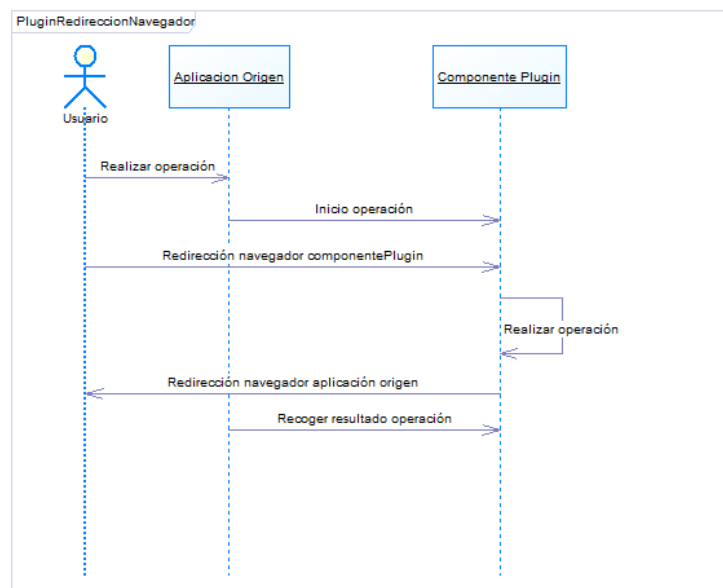
Básicamente se pueden distinguir 2 tipos de plugins:

- Plugins que implican exclusivamente comunicación entre aplicaciones (p.e. SISTRA2 y Regweb3), es decir, se invoca a servicios y se obtienen unos resultados.
- Plugins que implican comunicación entre aplicaciones y además redirección de navegador a nivel de usuario (p.e. SISTRA2 y Cl@ve), es decir, además de invocar servicios entre aplicaciones existe un “salto” del navegador del usuario al plugin para realizar una operación y al finalizar dicha operación un “retorno” a la aplicación origen.

A nivel general por motivos de reutilización y centralización de componentes, evitar acoplamiento entre aplicaciones y permitir versionado de APIs se propone el uso de componentes basados en servicios en lugar de definir APIs Java a incrustar en las aplicaciones que provoquen los problemas anteriores.

Estos componentes serán instalables en el servidor (EARs diferenciados) y proveerán un API basado en servicios versionables (SOAP, REST, ...). Se puede ofrecer un cliente de referencia de estas APIs a las aplicaciones consumidoras, pero si la aplicación consumidora quiere evitar dependencias podría consumir directamente esta API construyéndose su propio cliente.

Para plugins que requieran comunicación entre aplicaciones y además redirección de navegador a nivel de usuario se propone la redirección y recuperación de datos en base a tokens de un solo uso (OTP). El mecanismo propuesto es el siguiente:



- 1) Aplicación Origen invoca a servicio de Componente Plugin para iniciar la operación. En esta invocación se pasan los parámetros necesarios para la operación y como uno de los parámetros necesarios se le deberá indicar cuál es la url de callback a la Aplicación Origen tras finalizar la operación en el Componente Plugin. Como resultado de la invocación del servicio se obtendrá la url del Componente Plugin al que se debe redirigir el navegador para realizar la operación.
- 2) Redirección navegador de Aplicación Origen a Componente Plugin, que es dónde se realizará la operación.
- 3) Se realiza proceso en el Componente Plugin.
- 4) Una vez finalizada la operación, se redirige el navegador a la url de callback que la Aplicación Origen le indicó en el punto 1. Esta redirección se realiza pasando un token para que la la Aplicación Origen pueda recoger el resultado (por seguridad esta redirección debería ser por POST y HTTPS)
- 5) La Aplicación Origen recoge el resultado de la operación a partir del token invocando a Componente Plugin.

De esta forma la funcionalidad está aislada en un API simple basada en servicios y el intercambio de información, tanto en la llamada como en la respuesta, se realiza de forma segura, de aplicación a aplicación, a través de este API.

En la medida de lo posible estos componentes o plugins externos que ofrezcan una funcionalidad horizontal, podrían ser componentes reusables entre las distintas aplicaciones (p.e. Firmaweb, Cl@ve autenticación, Pasarela de pagos, etc.)



## 4. Esquemas de BBDD

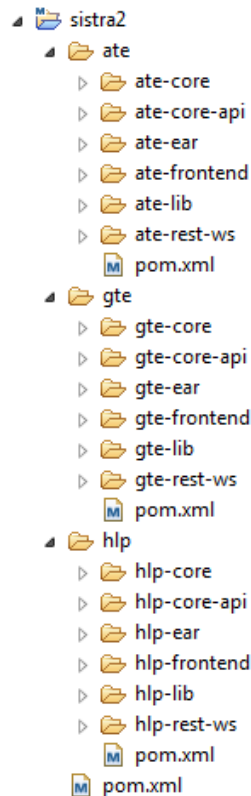
Los módulos que requerirán esquemas de BBDD serán:

- Gestor de Trámites (SISTRAGES): contiene las tablas de configuración del sistema y de la configuración y lógica de los trámites.
- Asistente de Tramitación (SISTRAMIT): contiene las tablas de persistencia de los trámites en curso por los ciudadanos y el registro de eventos de auditoría.

El módulo de Helpdesk (SISTRAHELP) en principio no requiere esquema de BBDD propio, ya que obtiene los datos necesarios a través del API de otros módulos (eventos auditoría del SISTRAMIT, estado de un pago a través del gestor de pagos correspondiente, etc.)

## 5. Arquetipo de la aplicación

El arquetipo de la aplicación está construido con Maven y dispone de un pom raíz que genera los 3 módulos de la aplicación (SISTRAGES, SISTRAMIT y SISTRHELP).



Para cada uno de ellos se distingue la siguiente estructura de módulos:

- xxx-core-api: contiene el api interno de la capa negocio de la aplicación, es decir, el interfaz entre la capa web (frontal web, servicios web rest/soap, etc.) y la capa de negocio. Básicamente se definen 3 packages:
  - services: interfaz de servicios ofrecidos
  - model: clases de modelo que se usan en los services para el intercambio de información
  - exception: lista de errores que puede generar la capa de negocio
- xxx-core: contiene la implementación de la capa de negocio. Distinguimos los siguientes packages:
  - ejb: capa EJB3 de implementa el facade a los services de negocio (implementa los interfaces de xxx-core-api). Estos servicios invocarían a los services definidos en Spring.
  - services: services de negocio en Spring (implementa los interfaces de xxx-core-api).
    - Distingue los siguientes packages

- component: componentes reusables dentro de la capa de negocio (services internos a la capa de negocio).
- repository: capa de acceso a la BBDD mediante JPA
  - interceptor: capa AOP que centraliza la auditoría y generación de logs
- xxx-frontend: interfaz web visual de la aplicación. Depende del módulo se emplea JSF/Primefaces (SISTRAGES y SISTRHELP) o Spring MVC/JSP (SISTRAMIT).
- xxx-rest-ws: interfaz web con servicios web tipo rest con Jersey. Se usará Swagger para la documentación del API. Si hiciese falta p.e. definir una capa SOAP con Apache CXF, se puede definir otro módulo (xxx-soap-ws). Dependiendo del uso de este api de servicios, si se va a consumir de forma externa a SISTRA2 habrá que definir versionado de estos servicios.
- xxx-ear / xxx-lib: módulos auxiliares que permiten el empaquetamiento de los ears/wars.

## 6. APIs integració

A nivel general se distinguen 2 tipos de APIs: una interna entre los módulos de SISTRA2 y otra para comunicarse con componentes externos. Ambas APIs estarán securizadas mediante un role de acceso específico (ver apartado de roles).

El API interna no requerirá versionado, ya que todos los módulos de SISTRA2 se evolucionarán dentro de una misma versión, por lo que no tiene sentido.

El API externa en cambio sí que tiene que disponer de versionado, para que los componentes externos que lo utilicen puedan ir evolucionando de una forma ordenada conforme se produzcan cambios en este API.

### 6.1. API SISTRAGES

#### API Interna

Será utilizada por SISTRAMIT para recuperar la configuración y definición de los trámites. Se distinguen los siguientes servicios:

Uri	Servicio	Parámetros	Resultado
/rest/interna/configuracionGlobal	Obtiene configuración global		Propiedades globales y plugins globales
/rest/interna/entidad/{id}	Obtiene configuración entidad	id: id entidad	Configuración entidad y plugins entidad
/rest/interna/tramite/{idTramite}/{version}/{idioma}	Obtiene definición trámite	idTramite: id trámite version:versión trámite idioma: idioma	Definición del trámite y configuración de los dominios utilizados en el trámite
/rest/interna/entidad/{id}/avisos	Obtiene avisos entidad	id: id entidad	Obtiene avisos activos de la entidad
/rest/interna/dominioFuenteDatos/{idDominio}	Consultar dominio de Fuente de Datos	idDominio: id dominio parametros: json con los parámetros del dominio (pasado por POST)	Resuelve un dominio de Fuente de Datos
/rest/interna/entidad/{id}/permisosHelpdesk	Consulta de los permisos de acceso al Helpdesk para una entidad. Usado desde Helpdesk para consultar	id: id entidad	Permisos de acceso al Helpdesk.

	permisos de acceso.		
--	---------------------	--	--

### API Externa

Servicios usados desde otras aplicaciones para consultar datos de definición de trámites (p.e. recuperar trámites electrónicos disponibles desde el Catalogo de procedimientos y servicios, para poder establecer un selector).

Uri	Servicio	Parámetros	Resultado
/rest/externa/invalidacion	Solicita una invalidación de datos		Propiedades globales y plugins globales

## 6.2. API SISTRAMIT

### API Interna

Será utilizada por SISTRAGES para solicitar invalidaciones de datos (configuración, trámites, dominios...) y por SISTRAHELP para recuperar los datos de persistencia y log de errores.

Uri	Servicio	Parámetros	Resultado
/rest/interna/invalidacion	Solicita una invalidación de datos		Propiedades globales y plugins globales

<< PENDIENTE ANALIZAR SERVICIOS REQUERIDOS POR SISTRAHELP>>

### API Externa

Será utilizada por componentes externos (Carpeta Ciudadana, Cuadro de Mando, ...) para obtener datos de tramitación.

<< PENDIENTE ANALIZAR SERVICIOS REQUERIDOS>>

## 7. Roles

Existen 2 niveles de control de seguridad:

- Nivel “grueso”: roles genéricos de la aplicación por los que están protegidos los EJBs.
- Nivel “fino”: roles que se pueden dar de alta en la aplicación que permiten especificar la seguridad por roles definidos dinámicamente. La gestión de estos roles dinámicos ya depende de cómo se quiera administrarlos.

A nivel general se distinguen los siguientes roles en la plataforma:

- STG\_SUP: Rol de superadministrador para administrar la configuración global de la plataforma.
- STG\_ADM: Rol de administrador de entidad, tendrá acceso tanto a la configuración de la entidad como al resto de elementos de una entidad. Para ser administrador de una entidad además de este rol necesitará el rol específico asignado en la configuración de una entidad para el administrador.
- STG\_DES: Rol de desarrollador de entidad. Para ser efectivo, además de este rol se deberán aplicar mediante la aplicación permisos sobre las áreas que se deseen (bien mediante un rol o directamente al usuario). Los permisos a aplicar sobre las áreas son:
  - Alta/Baja: permite realizar alta/baja sobre elementos del área (dominios, fuente datos, trámites...).
  - Modificación: permite crear y modificar versiones de trámite sobre el área.
  - Consulta: permite consultar los elementos del área.
  - Helpdesk: permite realizar consulta desde el módulo SISTRAHELP sobre los eventos de trámites del área.
  - Promocionar: permite exportar e importar versiones de trámites entre entornos.
- STG\_API: Rol de acceso al API Rest de SISTRAGES (es un rol para asignar a aplicaciones).
- STT\_API: Rol de acceso al API Rest de SISTRAMIT (es un rol para asignar a aplicaciones).

<< Hay que determinar si se separan los roles de acceso al API Rest dependiendo si se accede al API interno o externo >>

## 8. Infraestructura de soporte

La aplicación tendrá como infraestructura de soporte JBoss EAP 7 (Wildfly 10) y JDK 8.