Tecnologías

Librerías

Para poder acceder a la información es necesario tener conocimientos básicos de programación.

Se parte de https://confluence.um.es/confluence/pages/viewpage.action?pageId=662438145 que es una referencia a la tecnologías del SGI Hércules y del conocimiento de los proyectos de Spring Boot para back y de Angular para el front en el sentido de que ambos utilizan gestores de paquetes/dependencias que en un archivo te identifica todas las librerías que usa.

A continuación se proporcionan dos ejemplos de como se obtiene la información, uno de back y otro de front:

1.- En eGithub está el pom de csp https://github.com/HerculesCRUE/SGI/blob/main/sgi-csp-service/pom.xml

y en las dependencias están los artefactos que usa este servicio: commons-lang3, hibernate-jpamodelgen, hibernate-validator, …

2.- En los proyectos de angular las dependencias estan en el archivo package.json https://github.com/HerculesCRUE/SGI/blob/main/sgi-framework-angular/package.json

Por ejemplo:

"@angular/animations":~11.1.1",
"@angular/common": "~11.1.1",
"@angular/compiler": "~11.1.1",
"@angular/core": "~11.1.1",
"@angular/forms": "~11.1.1",
"@angular/platform-browser": "~11.1.1",
"@angular/platform-browser-dynamic": "~11.1.1",
"@angular/router": "~11.1.1",
"keycloak-js": "^11.0.0",
"rxjs": "^6.5.5",
"tslib": "^2.0.0",
"zone.js": "~0.10.2"

Conceptos principales

La filosofía del sistema está basada en tres conceptos principales:

- Principio de DATO ÚNICO: orientado a evitar duplicidad de esfuerzos, de fuentes de datos y mantener una mejor localización de la información.
- Principio de MODULARIDAD: el sistema se presenta como soporte para un conjunto de funcionalidades de diferente complejidad, con múltiples fases que pueden funcionar de forma autónoma.
- Principio de EVOLUCIÓN CONTINUA: en un entorno cambiante.

Nos basamos en los siguientes fundamentos:

- Los desarrollos llevados a cabo en este proyecto emplearán el patrón Modelo Vista Controlador (MVC) que separa los datos de una aplicación, la interfaz de usuario y la lógica de negocio en tres capas o componentes distintos

  Modelo: Esta es la representación específica de la información con la cual el sistema opera. Con el fin de independizarse del sistema de gestión de bases de datos (SGBD) subyacente y facilitar la implantación del sistema se utilizará un ORM (Object-Relational Mapping) como Hibernate.

De esta manera, se permite de forma transparente el uso de diferentes bases de datos (Oracle, PostgreSQL, SQLServer, …) como unidad de persistencia.

Vista: Esta presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario. Para la vista web se propone usar Angular (comúnmente llamado "Angular 2+" o "Angular 2"), es un framework para aplicaciones web desarrollado en TypeScript, de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página. Se libera así de carga al servidor ya que este solo se encarga de procesar peticiones JSON desde y hacia el explorador. El diseño del sistema se realizará de forma responsiva pudiendo verse de forma correcta en diferentes dispositivos y resoluciones, fijadas al inicio.

o Controlador: Este responde a eventos, usualmente acciones del usuario, e invoca peticiones al modelo y, probablemente, a la vista.

Arquitectura basada en microservicios, la cual permite distribuir software de calidad con mayor rapidez, reaccionando de una manera más rápida a cambios de negocio, saturación de ciertos servicios y minimizando posibles problemas.

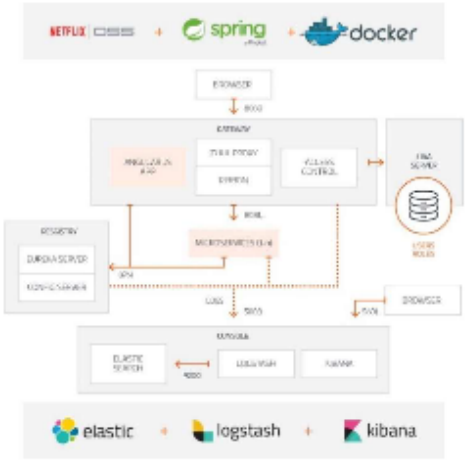
- La distribución del sistema se podrá realizar de dos maneras dependiendo de las necesidades y características de cada implantación (en ambos casos se dota al sistema de características escalables que le permiten soportar entornos en alta disponibilidad y altamente clusterizables):

o Como aplicación J2EE que se despliegue en un clúster de servidores de aplicaciones.

o Distribuida en contenedores utilizando tecnologías como Docker (para su ejecución) y Kubernetes (para el control y carga del sistema).

- Diseño basado en los principios de alta capacidad, alta disponibilidad y alto rendimiento, evitando tareas que requieran corte de servicio, permitiendo incorporación en caliente nuevos registros y procedimientos Facilidad de cambio, logrando la transparencia de la tecnología y, en la medida de lo posible, hacer uso de estándares industriales para interfaces clave.
- Uso de estándares abiertos, evitando formatos y protocolos privativos.
- Utilización de componentes y productos de terceros publicados bajo licencias Open Source y gratuitas.

En general la arquitectura de la aplicación propuesta a alto nivel seria la siguiente:



- Gateway – Puerta de entrada al sistema que se encarga de manejar el tráfico web del sistema, servir el

contenido estático de la aplicación Angular.

- Registry – Aplicación que se encarga de registrar la información de las aplicaciones (microservicios) existentes en el despliegue.
- Servidor de autenticación y autorización - UAA - se encarga de autenticar y autorizar a los usuarios en el sistema siguiendo un protocolo OAuth2.
- Microservicios – Son aplicaciones autocontenidas construidas con Spring Boot y dockerizadas que se encargan de manejar las peticiones de los clientes. Envían peticiones JSON y son manejados para lanzarse en paralelo para soportar carga extra.
- Con el fin de monitorizar, auditar y gestionar el sistema se utiliza el stack tecnológico ELK (ElastichSearch, Logstash y Kibana).