Máster Universitario de

Investigación en Ingeniería de Software y Sistemas Informáticos

Cod:31105151 Trabajo Final Máster

Estrategias para la integración de tecnologías RAD

Alumno: Gorka Sanz Lopategui

Tutor: José Antonio Cerrada Somolinos

Junio 2018-2019

Máster Universitario de Investigación en Ingeniería de Software y Sistemas Informáticos

Cod:31105151 Trabajo Final Máster

Estrategias para la integración de tecnologías RAD

Trabajo de Investigación

Alumno: Gorka Sanz Lopategui

Tutor: José Antonio Cerrada Somolinos

Contexto

Para el desarrollo de este trabajo nos situamos en el contexto de una propuesta por parte de un cliente. Con la idea de cubrir las necesidades de ese cliente se debe desarrollar una aplicación funcional. Usando cierto conocimiento previo adquirido por separado en las herramientas usadas en este trabajo de máster, se plantea el proyecto como un reto para profundizar en la integración y correcto funcionamiento de las mismas, con la única meta de que se coordinen como una única entidad perfectamente sincronizada, como un reloj.

La aplicación sirve como marco para trabajar las cuestiones que aquí se desarrollaran. El resultado es una aplicación para la gestión y recordatorio de actividades relacionadas con el ámbito educativo.

Se usará como base para demostrar de forma práctica los conceptos aquí descritos. El producto final consiste en una aplicación que permita organizar la gestión de los buenos días dentro del centro educativo.

Los buenos días son una actividad en la cual, una persona comunica contenido por los micrófonos del centro, contando aquello que desea relatar, informar, transmitir y que ha preparado previamente, siguiendo una temática propuesta desde el propio centro.

La elección de la persona que le toca impartirlos, es aleatoria, y puede ser un profesor, un alumno, u otro personal del centro. Aunque el proceso es aleatorio, no todos los profesores participan y los que participan indican previamente que días de la semana pueden realizarla (debido a los horarios), y cuantas veces, los usuarios quieren participar a lo largo del año.

El calendario se generará aleatoriamente, teniendo en cuenta los festivos y las preferencias de cada profesor, y se notificará al profesor mediante correo electrónico, un día antes de que le toca impartir el buenos días.

Una parte importante de la aplicación, y que se debe investigar, y cuyo resultado se va a plasmar en esta memoria, es el uso de la autenticación dentro de la aplicación usando Oauth2 a través de las cuentas de google corporativas que el centro posee para cada profesor.

Objetivos del proyecto.

Con este proyecto se pretende ahondar en aquellas herramientas que permiten construir un software de manera segura. Añadiendo funciones robustas, claras y sobretodo reutilizables en otras partes de la aplicación, y/o en otros productos que se deseen desarrollar a posteriori, marcando un punto de anclaje sólido para el desarrollo de arquitecturas escalables, capaces de atender a la demanda de múltiples usuarios.

Profundizando más en esta idea podemos desglosar este desarrollo y para la consecución de los siguientes objetivos podemos enumerar los siguientes puntos clave:

1. Utilización del sistema de autorización del usuario mediante sí el uso de Single Sign On.
2. Integración del framework que se enfoque en la seguridad y en la autenticación con el framework Apache Wicket.
3. Creación de tareas en segundo plano desatendidas y sin necesidad de interacción por parte del usuario.
4. Explorar las técnicas que permitan un desarrollo rápido de la aplicación.
5. Extraer información desde orígenes de terceros.

Debido a la integración de todas estas tecnologías tan punteras ha sido necesario adaptar componentes, para hacerlos compatibles con el resto del ecosistema. Todo ello se describe en el apartado correspondiente dentro de la memoria.

Estado del Arte

SSO vs Contraseñas

Uno de los grandes problemas con las aplicaciones ha sido siempre como verificar el usuario que accede a una aplicación, si es realmente quien dice ser, si tiene realmente los permisos sobre aquello que desea acceder, que no puede acceder a partes que no necesita.

El internet de hoy en día no es como lo que existía antes del 2.0, antes de una World Wide Web, donde el usuario es el centro y a donde este usuario se le da la posibilidad de crear contenido, así como adaptar las preferencia de lo que necesita y desea encontrar.

Hoy en día internet, incluso para el más mínimo servicio alojado, se necesita crear un perfil sobre cada uno de nosotros, en principio con la sana intención de ofrecer mejores respuesta y resultados que se ajusten de una forma notable. Todo con la idea de mejorar lo que ocurría antaño.

Hoy en día vivimos en un internet de servicios, donde conviven muchísimas aplicaciones, de las cuales algunas han llegado hasta nuestros días y otras se han quedado por el camino. Vivimos en un internet donde pedimos a cada una de las personas del ciberespacio que especifique sus preferencias, el contenido que crear y una cantidad ingente de información es guardada. Una información, que el usuario, como si fuera la partida de su videojuego favorito, necesita recuperar para continuar evolucionando.

Si se piensa como un videojuego, el usuario necesita almacenar su información, guardada a buen recaudo, y no deseamos que otro usuario pueda ver nuestro progreso, el nivel en que estamos, y los objetos que hemos conseguido. Este progreso, en los videojuegos no es tanto problema, ya que podemos guardar la partida en una tarjeta de memoria y si queremos continuar la partida en otro sitio, simplemente la portamos con nosotros y la conectamos en la nueva ubicación física para volver a tener nuestro mundo tal y como lo habíamos dejado la última vez.

En los videojuegos podemos funcionar de esa manera, ya que, los valiosos datos no están guardados en ningún servidor lejano que pueda estar caído, para sustraer nuestra identidad es necesario robarnos presencialmente esa tarjeta de memoria.

Con el surgimiento de las aplicaciones web, por espacio, y por comodidad se ha visto inviable guardar una tarjeta de memoria por cada servicio que usamos. Ahora la información se guarda en la propia internet o como a la mayoría les gusta llamar, la nube. El hecho de guardar

la información de esta forma ha provocado que para recuperar la información el servidor que la contiene, debe estar completamente seguro de a quién se la manda.

Históricamente, las páginas web han permitido crear usuarios dentro de su plataforma, todo esto gracias al uso de elementos como las bases de datos y lenguajes de programación de alto nivel que pueden hacer consultas sobre ella. El primer tipo de sistema de validación de usuario ha sido mediante la tupla usuario-contraseña. Este mecanismo permite a un usuario darse de alta en un servicio ofreciendo la mínima información posible sobre su identidad.

Con el crecimiento de los servicios, el usuario cada vez tenía que registrarse en nuevos y diferentes sitios para obtener servicios complementarios, los cuales en muchas ocasiones solo necesitaba utilizar, una única vez y después jamás volvía a utilizar. Esto provocaba que frecuentemente, el usuario olvide su contraseña con el tiempo o que acabará utilizando la misma contraseña para todos los sitios web. Dado que unos sitios son más seguros que otros, si averiguan tu contraseña en un sitio, para un hacker, solo era cuestión de probar suerte con la misma tupla en otros servicios.

El mecanismo de usuario contraseña ha ido evolucionando con el tiempo, primero al no guardar las contraseñas directamente en la base de datos, sino, que se guarda la contraseña cifrada(su hash), de tal forma que, aunque un usuario malintencionado consiga acceder a la información le resulte impracticable hacer uso para fines perjudiciales.

Con el tiempo se exigió que este usuario tuviera asociado una cuenta de correo electrónico, con el fin de evitar usuario duplicados en la base de datos y, además en caso de que el usuario olvide la contraseña del servicio, pudiera restaurar y crear una nueva gracias a ese cuenta de correo electrónico.

El Captcha

Consiste en una prueba controlada por una máquina, en la misma línea que la Prueba del test Turing, permite distinguir entre ordenadores y personas. Se trata así de una prueba para verificar si quién está delante del ordenador, es un humano, ya que una máquina no podría interpretar la secuencia correcta.

Su finalidad está clara, impedir que los robots (spambots) puedan lograr acceso automático a determinados servicios online. De tal manera, que no puedan participar en foros o utilizar cuentas de correo para, para por ejemplo, realizar el envío de spam o correo basura.

Estos mecanismos indicados arriba no tienen más finalidad que ayudar a proteger la seguridad del usuario. Son mecanismos que hoy en día vemos de forma habitual, pero hace

unos años nadie creía necesarios y los usuarios creían rechazaban pensando que no era nada más que un sistema engorroso y artificial.

A medida que la importancia de las redes sociales crecía y los usuarios cada vez publican más información privada y personal, ellos mismo se daban cuenta de la importancia de tener sus cuentas bien protegidas ante usurpaciones y robos de identidad. Los usuarios abrazaron con las manos abiertas estas soluciones “double check”, ya que el aporte extra de protección frente al coste de usarlas es importante, eso sin tener en cuenta los posibles riesgos asociados.

A continuación se va describir más profundamente, aquellos elementos que han sido claves en el desarrollo de la aplicación, se comenzará describiendo la arquitectura, para posteriormente nombrar aquellos elementos clave necesarios y que fundamentan el sentido sobre el que se apoya este trabajo, justificando cada uno de ellos y ofreciendo una comparativa con otras soluciones del mercado. Por cada artefacto se comentará lo destacable de sus características y sus puntos fuertes. En caso de encontrarse alguna debilidad que descarte o no recomiende su uso bajo determinadas circunstancias, también se indicará.

Arquitectura

Para la arquitectura de la aplicación se ha decidido seguir un diseño DDD (Domain Drive Design) ý siguiendo pauta sugeridas por Eric Evans en su libro Domain-Driven Design: Tackling Complexity in the Heart of Software.

A la luz de las ideas recopiladas en este libro, Spring propone un diseño basado en componentes reutilizables allí donde se desee, mediante la Inyección de dependencias.

Al hacer uso de spring, se ha decidido utilizar una arquitectura de cebolla, donde hay capas sobre capas envolviendose las unas a las otras.

Arquitectura de la cebolla: hay varias arquitecturas tradicionales, como la arquitectura de 3 niveles y la arquitectura de n niveles, todas con su propias ventajas y desventajas. Todas estas arquitecturas tradicionales tienen algunos problemas fundamentales, tales como: un acoplamiento estrecho y preocupación por la no separación.

El Modelo-Vista-Controlador es la arquitectura más utilizada en estos días. Resuelve el problema de la preocupación por la separación, ya que existe una separación entre la IU, la lógica empresarial y la lógica de acceso a los datos. MVC resuelve el problema de la separación de la preocupación, pero el problema del acoplamiento alto aún permanece.

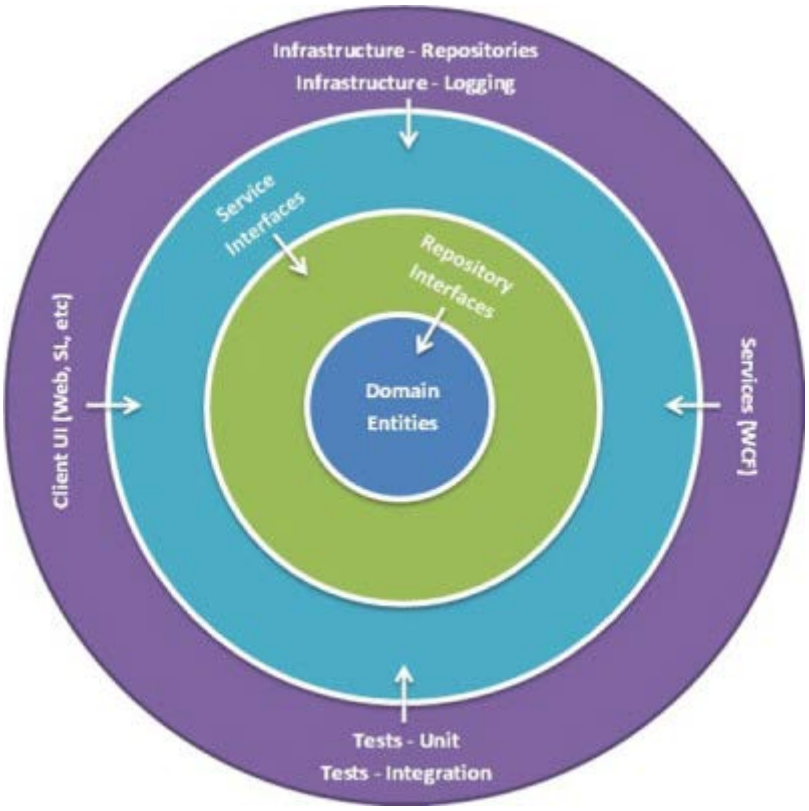
Alto Acoplamiento: Cuando una clase depende de una dependencia concreta, se dice que está estrechamente relacionada con esa clase. Un objeto estrechamente acoplado depende de otro objeto; eso significa cambiar un objeto en una aplicación estrechamente acoplada, a menudo requiere cambios en una cantidad de otros objetos. No es difícil de hacer cuando una aplicación es pequeña, pero en una aplicación de nivel empresarial, es demasiado complejo hacer tantos cambios.

Bajo acoplamiento: Significa que dos objetos son independientes y un objeto puede usar con otro objeto sin depender de él. Es un objetivo de diseño que busca reducir las interdependencias entre los componentes de un sistema con el objetivo de reducir el riesgo de que los cambios en un componente requieran cambios en cualquier otro componente.

Por otro lado, “Onion Architecture” aborda tanto la preocupación por la separación de componentes como los problemas del alto acoplamiento. La filosofía general de la arquitectura de cebolla es mantener la lógica de negocios, la lógica de acceso a los datos y el modelo de la aplicación separados. Implica empujar las dependencias lo más hacia fuera posible del centro, y significa que todos se acoplan hacia el centro.

Ventajas de la arquitectura de cebolla.

- Proporciona una mejor capacidad de mantenimiento, ya que todas las partes de código dependen de las capas internas o del centro.
- Proporciona una mejor capacidad para realizar pruebas, ya que la prueba de la unidad se puede crear para cada una de las capas separadas, sin el efecto colateral sobre otros módulos de la aplicación.
- Permite desarrollar una aplicación con un acoplamiento bajo, ya que la capa externa de la aplicación, siempre se comunica con la capa interna a través de interfaces.
- Las entidades de dominio son parte central. Puede tener acceso tanto a la base de datos como a las capas de UI.
- Las capas internas nunca dependen de la capa externa. El código más propenso a sufrir debe ser parte de la capa externa antes que la interna. La arquitectura de la cebolla se basa en gran medida en el principio de inversión de dependencia. La interfaz de usuario se comunica con la lógica empresarial a través de interfaces.



Tiene cuatro capas.

1. Capa de entidades de dominio
2. Capa de repositorio

3. Capa de servicio
4. Capa de UI (aplicación web / API web / proyecto de prueba de unidad)

Capa de entidades de dominio: es la parte central de la arquitectura. Contiene todos los objetos de dominio de aplicación. Si una aplicación se desarrolla con el marco de la entidad ORM, esta capa es la que contiene clases de POCO (Code First) u Edmx (Database First) con entidades. Estas entidades de dominio no tienen ninguna dependencia.

Capa de repositorio: la capa se proporciona para crear una capa de abstracción entre la capa de entidades de dominio y la capa de lógica de negocios de una aplicación. Usando un patrón de acceso a datos, se ofrece un enfoque de acceso a los datos más débilmente acoplados.

Creamos un repositorio genérico, que consulta el origen de datos para obtener la información, asigna los datos del origen de datos a una entidad de negocio y persiste los cambios en la entidad de negocio hacia el origen de datos.

Capa de servicio: la capa contiene interfaces que se utilizan para comunicarse entre la capa de IU y la capa de repositorio. Mantiene la lógica de negocios para una entidad, por lo que también se denomina capa de lógica de negocios.

Capa UI - Es la capa más externa. Podría ser la aplicación web, la API web o el módulo de prueba unitaria. Esta capa tiene una implementación del Principio de Inversión de Dependencia, para que la aplicación se construya como una aplicación acoplada libremente. Se comunica a la capa interna a través de interfaces.

Se ha decidido usar SSO sobre otros sistemas debido a que aporta seguridad, implica subsanar gran parte de los problemas existentes de tecnologías anteriores como el uso de contraseñas. La contraseña del usuario en ningún momento se encuentra almacenada internamente dentro de esta plataforma desarrollada. Así el usuario puede usar una misma contraseña a través de muchos otros servicios alojados en diferentes sitios de internet.

Hoy en día el usuario por comodidad ya está acostumbrado a usar servicios de terceros (Google, Facebook, OpenId) para demostrar su identidad, para el usuario es más cómodo ya que no debe recibir nunca más el correo de verificación al crear una nueva cuenta.

Valoración de candidatos

Para obtener la autorización del usuario mediante el uso de single sign on de Google. se han evaluado los siguientes candidatos.

- google-auth-library-nodejs 

  Como ventajas principales encontramos que es una librería apoyada por google.

  Como inconveniente, necesita usarse junto con la tecnología nodeJs y usar por tanto un lenguaje compatible con Javascript.

- Spring security

  Bajo desarrollo activo. tiene mucho más apoyo de la comunidad. La seguridad de Spring tiene extensiones que brindan soporte tanto para Oauth, y kerberos, y SAML.

Para realizar la Integración del un framework de seguridad con la librería Apache wicket. Se han sopesado las siguientes opciones como candidatos:

- Apache Shiro

  La autenticación es simple e intuitiva por diseño. El proceso se basa en crear áreas y lo realiza el desarrollador utilizando solo unas pocas llamadas a métodos. El diagnóstico de errores es facilitado por una rica jerarquía de excepciones. La funcionalidad de Remember-me está incluida per se.

  Los DAO son enchufables se pueden usar para implementar realms y es posible iniciar sesión de un sujeto en múltiples realms mientras se mantiene una visión global uniforme.

  La autorización también está basada en el sujeto. Los derechos de acceso están determinados por los roles y permisos de los individuos. Para comenzar a implementar los permisos de inmediato, uno puede usar la sintaxis de permiso con comodines propia de Shiro.

  Para mejorar la experiencia del usuario y el rendimiento, se admiten diferentes soluciones de almacenamiento en caché.

Por contra, no soporta Saml o Oauth, No hace mención de apoyar las políticas de seguridad before y after, el desarrollo activo parece limitado, el sitio web todavía contiene información obsoleta.

- Spring security

  Es un marco que permite a un programador imponer restricciones de seguridad a las aplicaciones web basadas en Spring-framework a través de componentes JEE. En resumen, es una biblioteca que se puede usar y ampliar para personalizar según las necesidades del programador.

  Su principal área de operación es controlar la autenticación y la autorización en el nivel de solicitud web, así como el nivel de invocación del método. Quizás, la mayor ventaja de este marco es que es muy potente y altamente personalizable en su implementación.

  Aunque sigue la idea de usar la convención sobre la configuración, los programadores pueden elegir entre las disposiciones predeterminadas o personalizarlas según sus necesidades.

Para el objetivo tener tareas en segundo plano desatendidas y sin necesidad de interacción por parte del usuario, se han valorado las siguientes tecnologías.

- Quartz

  Quartz es un framework open source, con licencia Apache 2.0 para la planificación y gestión de tareas. Con planificación flexible de tareas, mantenimiento del estado de las tareas incluso en caso de fallos y reinicios de máquinas, capaz de participar en transacciones JTA. Y además nos proporciona una completo API, con muchas clases de utilidad y muchos tipos de escuchadores.

  Por contra Quartz es deficiente en su conjunto global de características. Es solo una biblioteca de código para la ejecución de trabajos. No hay una consola de monitoreo para revisar los errores y el historial, ningún registro útil, no es compatible con múltiples nodos de ejecución sin ayuda extra, no hay una interfaz de administración, no hay alertas ni notificaciones, además usa mecanismos inflexibles ante los errores de recuperación de trabajos fallidos causando trabajos perdidos.

Quartz proporciona soporte adicional para varios nodos, pero requiere una configuración avanzada añadida. Quartz también proporciona un complemento llamado Quartz Manager, que también necesita de una configuración avanzada adicional, es una aplicación flash y es increíblemente engorroso y poco práctico de usar.

- Spring scheduling tasks

  Spring Scheduler es una capa de abstracción escrita para ocultar las implementaciones de Executors a través de diferentes JDK que tienen sus propias implementaciones específicas.

  La ventaja de usar Spring Scheduler sin usar directamente las clases específicas de Quartz Scheduler es que la capa de abstracción proporciona flexibilidad y acoplamiento bajo.

- Async Tasks

  Esta es una pequeña extensión de Wicket que permite la administración de tareas asíncronas (iniciar, interrumpir y reiniciar) y mostrar el progreso y el estado de una tarea al usuario. Para permitir que las tareas comuniquen su progreso, deben implementar IProgressObservableRunnable.

  Además, las tareas pueden administrarse fuera del alcance de una página web sin romper el contrato de Wicket de que todos los Componentes deben ser serializables, ofreciendo la interfaz de TaskManager. Se proporciona una implementación predeterminada de dicho TaskManager.

  Siempre que se mantengan las tareas en segundo plano libres de referencias a sus clases de Wicket, no habrá problemas. Hay que tener especial cuidado con usar referencias a beans de spring dentro de las tareas.

Para conseguir un rápido desarrollo de la aplicación. Se ha planteado el uso de las siguientes tecnologías desde los dos puntos de vista de la arquitectura.

- Para la integración rápida de los componentes a nivel de artefacto se han valorado las siguientes tecnologías.

    - Ant con ivy

      Ant fue la primera entre las herramientas de construcción "modernas". En muchos aspectos es similar al Make. Tiene una curva de aprendizaje inicial muy baja, lo que permite que cualquiera pueda comenzar a usarlo sin ninguna preparación especial. Se basa en la idea de programación de procedimientos.

      Para definir su estructura de programación usa notación XML. Al ser XML de naturaleza jerárquica, no es adecuado para el enfoque de programación de procedimientos. El XML tiende a volverse inmanejable, denso, grande, incluso cuando se usa con proyectos muy pequeños.

    - Maven

      Maven continúa usando XML como el formato para escribir la especificación de compilación. Sin embargo, la estructura es diametralmente diferente.

      Maven se basa en convenciones y proporciona los objetivos disponibles (targets) que se pueden invocar. Como la característica adicional, y probablemente la más importante, Maven introdujo la capacidad de descargar dependencias a través de la red (la cual adoptó Ant a través de Ivy). Revolucionó la forma en que entregamos el software.

      Sin embargo, Maven tiene sus propios problemas. La administración de dependencias no maneja los conflictos entre diferentes versiones de la misma biblioteca (algo en lo que Ivy es mucho mejor).

      XML como el formato de configuración de construcción, está estrictamente estructurado y altamente estandarizado. La personalización de los objetivos es difícil.

      Como Maven se centra principalmente en la administración de dependencias, los scripts de compilación complejos y personalizados son realmente más difíciles de escribir en Maven que en Ant.

- Gradle

  Gradle no utiliza XML. En su lugar, tenía su propio DSL basado en Groovy (uno de los lenguajes JVM). Como resultado, los scripts de compilación de Gradle tienden a ser mucho más cortos y claros que los escritos para Ant o Maven. Gradle está diseñado para llevar el software a lo largo de su ciclo de vida, desde la compilación hasta el análisis estático y las pruebas hasta el empaquetado y la implementación.

  El primer problema es que eres totalmente libre de hacer lo que quieras: no hay un diseño estándar para los ficheros fuente, no hay un ciclo de vida de compilación estándar.

  Así que volviendo a los viejos tiempos de Ant: cualquier desarrollador que venga de un proyecto Gradle a otro debe aprender todo desde cero, ya que las dos rutas de compilación y las estructuras de proyecto son totalmente diferentes. Sin comentar que el programador debe aprender un nuevo lenguaje.

- Para la integración rápida de los componentes a nivel de clase se han valorado las siguientes tecnologías.

  - Spring Boot

    Spring tiene un gran ecosistema en torno al marco central, y evoluciona más rápido que Java EE, lo cual es natural como estándar y acaba incluyendo algunos puntos en común establecidos, mientras que otros son solo propios de spring.

    Por lo tanto, el ecosistema de Spring está más evolucionado con respecto a las tecnologías de nube modernas, como las bases de datos no relacionales, ciertos tipos de sistemas de mensajería o tecnologías de descubrimiento de servicios.

  - Java EE

Java EE ofrece algo más de estabilidad para los desarrolladores, y aunque tarda más en evolucionar se aprovecha de la madurez de las mejores ideas existentes en el mercado.

Por contra históricamente los servidores Java EE tardan más en arrancar, debido a todo el entramado extra que debían incluir para cumplir el perfil Java EE.

Para extraer información desde orígenes de terceros en la web. existen las siguientes alternativas que se deben valorar.

- JSoup

  Permite extraer y analizar HTML desde una URL, archivo o cadena de texto en la cual encuentra y extrae datos, haciendo búsquedas transversales DOM o selectores de CSS. Permite la manipulación de los elementos HTML, atributos y texto. Limpiar el contenido enviado por el usuario usando una lista blanca segura, para evitar ataques XSS. Ordenar el HTML de salida.

  Por contra jsoup no forma parte del estándar java y debe ser añadida como una dependencia a nuestro proyecto añadiendo peso a la suma de tamaño total del proyecto.

- Jaunt

  Es una librería que permite la prospección web y extracción de datos en JSON, así como trabajar con formularios y tablas. Controlar / procesar solicitudes / respuestas HTTP individuales.Interactuar con interfaces API REST o aplicaciones web usando JSON, HTML, XHTML, or XML.

  Tiene una versión que no es gratuita y de dos años, mucho más restrictiva que la licencia MIT, ya que NO permite la redistribución del archivo jar. La versión empresarial es una licencia comercial (que generalmente cubre un número ilimitado de empleados) cuyos términos son completamente negociables, por lo que puede o no, permitir la redistribución, según las necesidades del cliente.

- HtmlUnit

  Es una biblioteca java de código abierto para crear llamadas HTTP que imita la funcionalidad del navegador. HtmlUnit se utiliza principalmente para pruebas de integración en marcos de prueba de unidades como JUnit o TestNG. Esto se hace solicitando páginas web y haciendo aserciones sobre los resultados.

A continuación se muestra un cuadro comparativo con las principales fortalezas de cada una de ellas,

JSoup	Jaunt	HtmlUnit
Es un parser de HTML		Simula internamente un navegador
Licencia Free	Licencia comercial	Licencia Free
Soporta selectores XPath y Css	Se integra muy bien con servicios Rest y JSon	Se integra muy bien con Selenium
	Usa sintaxis propia	Necesita software adicional

Justificación de las herramientas utilizadas

Spring Boot

Spring es muy popular por varias razones. El enfoque de inyección de dependencias de Spring fomenta la escritura de código fácil de testear. Fácil de usar, pero con poderosas capacidades para gestión de transacciones en base de datos. Spring simplifica la integración con otros marcos de Java como JPA / Hibernate ORM, Struts / JSF / etcétera.

Spring Boot se basa en el Spring Framework. El Spring Framework se desarrolló hace más de quince años como una alternativa ligera a la pila Java Enterprise. Permite a los desarrolladores crear aplicaciones de nivel empresarial sin el peso de la pila JEE tradicional.

Los módulos centrales de los ecosistemas de Spring son estables durante mucho tiempo, y la mayoría de los cambios son compatibles con versiones anteriores. Los cambios de última hora se anuncian mucho antes del lanzamiento, y generalmente hay una hoja de ruta para la migración sin contratiempos disponible a tiempo.

Spring Boot y los principales módulos Spring en el ecosistema son software de código abierto. Sin embargo, Pivotal, una empresa que ofrece plataformas y herramientas para crear un mejor software, es mantiene en gran medida el proyecto y lo respalda. Spring es una piedra angular de su negocio. Su uso es gratuito y probablemente se mantendrá durante otra década.

Con Spring se puede conectar fácilmente la aplicación a bases de datos relacionales, bases de datos NoSQL o servicios de cola de mensajería. Todo ello a lo largo de un camino sin excesivas complicaciones.

Es compatible con Oracle, PostgreSQL, MySQL, MongoDB, Redis, Solr, ElasticSearch, Rabbit MQ, ActiveMQ y muchos, muchos más.

Con Spring Boot puede desarrollar aplicaciones web únicamente del lado del servidor, RESTful y otras API web, o incluso crear trabajos por lotes y/o aplicaciones de línea de comandos regulares.

Los desarrolladores aman la programación con Spring Boot. Se sienten mucho más productivos, disfrutan de los beneficios del ecosistema de Spring y de la tranquilidad de tener que ejecutar este sistema en producción. Cuando los desarrolladores están encantados con su pila de tecnología, generalmente son más productivos. Spring ayuda a tener el trabajo farragoso terminado y más rápido que sin él.

Un problema habitual con el software de código abierto son los recursos, como la documentación y los talentos del mercado. Con Spring, se encuentran muchos libros, videos y cursos de capacitación disponibles de manera gratuita o a bajo coste.

Apache Wicket

Con Wicket las páginas web son objetos: las páginas web no son solo archivos de texto enviados al cliente. Son instancias de objetos y podemos aprovechar la POO para diseñar páginas web y sus componentes. Con Wicket también podemos aplicar la herencia al marcado HTML para construir un diseño gráfico consistente para nuestras aplicaciones

A veces, los requisitos y las especificaciones comerciales para un proyecto cambian, o nos gustaría mejorar nuestro código a medida que avanzamos en el proyecto, por lo que refactorizar el código debería ser fácil en un marco web (para muchos no lo es). Debido a la naturaleza de los componentes y de los objetos Java, implica que la refactorización no es muy diferente de lo que haría para una aplicación Java normal. Esto significa que tenemos con este framework un tiempo de respuesta bajo para los cambios de los requisitos.

Es un marco basado en componentes con buenos conceptos que atraen a los desarrolladores. Pero lo mejor es que Wicket realmente se siente como un verdadero marco web orientado a objetos. Todos sus componentes, páginas, paneles son objetos Java (con un fichero html) .

Cada componente web extiende de una clase componente base y, a su vez, puede ampliarse. Otra forma de crear componentes en Wicket es a través de la composición de otros componentes web dentro de un Panel.

Permite empaquetar sus componentes dentro de un archivo jar y llevarlos a otro proyecto, e idealmente, podría hasta surgir una biblioteca común de componentes para la organización. Una organización existente con componentes genéricos de código abierto es el proyecto Wicket Stuff.

Wicket también viene con validadores y convertidores integrados o interfaces para implementar los propios.

Wicket cultiva un buen patrón arquitectónico al imponer que cada componente esté respaldado por un modelo. Es el Modelo para el Componente el que maneja la recuperación de datos. El mismo componente se puede usar con diferentes implementaciones de modelos que proporcionan datos de diferentes fuentes o formas.

Se pueden construir y re-utilizar modelos genéricos que ya saben cómo recuperar, por ejemplo, a través de Spring-Data, una entidad por ID. Wicket proporciona una serie de modelos predefinidos. Una buena comprensión de qué son los modelos y cómo / cuándo deben usarse cada uno, es una cosa muy importante en la que prestar especial atención a la hora de aprender Wicket.

Los comportamientos son otro concepto que se ha separado del componente. La razón de esto es que incluso la forma en que un componente se "comporta" puede ser similar al comportamiento de otro componente y puede ser un candidato para reutilización o extensión. AjaxBehavior es un ejemplo tipo de esos comportamientos de componentes cruzados.

En Wicket también se pueden usar Visitantes (según el patrón de Visitor) para la comunicación transversal de componentes en una Página. Además, Wicket proporciona un mecanismo EventBus a través del cual los componentes pueden comunicarse.

Debido a que Wicket ha sido diseñado desde el principio como un proyecto estable, permite pasar objetos Java de una instancia de página a otra. Además de la intención más definida, es la forma más fácil de seguir la pista de qué páginas utilizan un modelo en concreto, por otro lado esto también puede verse como un plus de seguridad adquirido.

Vale la pena mencionar la buena integración con los marcos de inyección de dependencias como Spring / Guice. Se puede inyectar fácilmente Spring beans dentro de cualquiera de sus páginas, paneles, componentes, modelos proporcionando sólo la anotación @SpringBean.

Wicket es compatible con la gestión configurable de la versión de la página. Cuando los usuarios envían un formulario o siguen un enlace desde una página a la que accedieron con el botón de atrás en el navegador, Wicket puede revertir el objeto de la página al estado en que se encontraba cuando la página se procesó originalmente. Esto significa que puede escribir aplicaciones web que admiten el botón Atrás, con muy poco trabajo.

Wicket proporciona una forma fácil de escribir una aplicación que admite el uso de ventanas múltiples y pestañas, lo que permite al desarrollador reaccionar correctamente cuando los usuarios abren una nueva ventana o pestaña del navegador.

Wicket es 100% de código abierto: Wicket es un proyecto superior de Apache y no depende de ninguna compañía privada. Wicket siempre se lanzará bajo la licencia Apache 2.0 y estará disponible gratuitamente

Wicket es un proyecto impulsado por la comunidad: el equipo de Wicket apoya y promueve el diálogo con los usuarios a través de dos listas de correo (una para usuarios y otra para

desarrolladores) y un Apache JIRA (el sistema de seguimiento de incidencias). Además, como cualquier otro proyecto de Apache, Wicket se desarrolla prestando gran atención a los comentarios de los usuarios y a las funciones sugeridas.

Apache Maven

Permite la configuración rápida del proyecto, sin archivos build.xml complicados, solo un POM y listo.

Todos los desarrolladores en un proyecto usan las mismas dependencias jar debido al POM centralizado.

Permite obtener varios informes y métricas para un proyecto, reducir el tamaño de los entregables, ya que los jar se pueden descargar desde internet.

Promueve el diseño modular de código. al simplificar la gestión de proyectos múltiples, permite que el diseño se distribuya en varias partes lógicas, tejiendo estas partes mediante el uso de dependencias en archivos pom.

La gestión de la dependencia está claramente delineada con el mecanismo de administración de dependencias, No existe ninguno de los problemas clásicos de conocer qué versión de librerías debe ser usada.

Ciclo de vida de sólido: hay un fuerte ciclo de vida definido por cual un sistema de software se inicia desde el comienzo de una construcción hasta el final. Esto tiene el beneficio adicional de permitir que las personas se muevan de un proyecto a otro y hablen utilizando el mismo vocabulario en términos de desarrollo de software.

Maven está avanzando rápidamente y existe la posibilidad de contar con muchas herramientas de alto valor en torno a Maven (CI, proyecto de Dashboard, integración de IDE, etcétera).

JavaMail

El diseño de los paquetes y las clases en la API de Java Mail demuestra uno de los objetivos principales de sus diseñadores: que el nivel de esfuerzo requerido por el desarrollador para construir una aplicación debe ser dictado por la complejidad de la aplicación y el nivel de control requerido por el desarrollador para la aplicación. En otras palabras, mantenga la API

lo más simple posible. La aplicación contextualizada en este trabajo y los ejemplos que se incluyen con la API de Java Mail demuestran ampliamente este punto.

A primera vista, la cantidad de clases de API de Java Mail y el diseño detallado de estas clases, pueden hacer creer que enfrente espera una curva de aprendizaje pesada. Pero en realidad, una vez se comienza a trabajar, se descubre que esta API es una herramienta simple y práctica para implementar una funcionalidad robusta de correo en las aplicaciones.

La principal diferencia entre usar SMTP / IMAP vs API de Gmail. sería la forma de iniciar sesión, SMTP e IMAP permiten la autenticación mediante el inicio de sesión del cliente (inicio de sesión y contraseña), mientras que la API de Gmail requerirá que se use la autenticación abierta (Oauth2) para ser exactos.

Ahora, hay una serie de inconvenientes a eso. Así que ahora, si el usuario cambia su contraseña, ninguna solución funcionará hasta que el usuario actualice la contraseña en la aplicación.

FullCalendar2

La versión 2.0 de FullCalendar se integra perfectamente junto con wicket y ofrece enormes mejoras con respecto a las zonas horarias, la internacionalización y la manipulación de fechas, así como la configuración del componente desde el lado del servidor. Las llamadas al componente javascript se realizan a través de eventos ajax siendo también capaz de responder a estas peticiones mediante ajax.

Integración con el Single Sign On

Esta es sin duda una de las piedras angulares más importantes del proyecto y por ello se encuentra descrita la primera. El desarrollo del proyecto tenía como uno de los requisitos principales el hecho de poder autenticarnos dentro de la aplicación mediante el uso de cuentas de correo corporativas otorgadas por el entorno empresarial de Google. SSO podemos ver todos los actores involucrados en la petición.

Para el desarrollo de la aplicación se ha prestado más atención a la integración Google. La tecnología utilizada podría permitir también una integración rápida mediante oauth2 con Facebook. El uso y la exploración de otros mecanismos de autenticación oauth2 como Open Id, Octa y parecidos queda fuera del ámbito de investigación de este proyecto.

El mecanismo de autenticación Oauth permite conocer y comprobar la autenticidad de una persona mediante el uso de un servicio externo bien conocido, en este caso mediante Google. Ya que el registro se va a hacer en los servidores de Google, así como su validación. Lo que tiene obtiene nuestra aplicación es una respuesta en forma de token validando los credenciales obtenidos en este servicio, la integración no ha sido sencilla ya que la información disponible en la red es muy escueta.

Para obtener las credenciales de la autenticación oauth2 de Google hay que entrar en la sección de la consola API de Google y dentro de esta sección hay que hacer clic en credenciales.

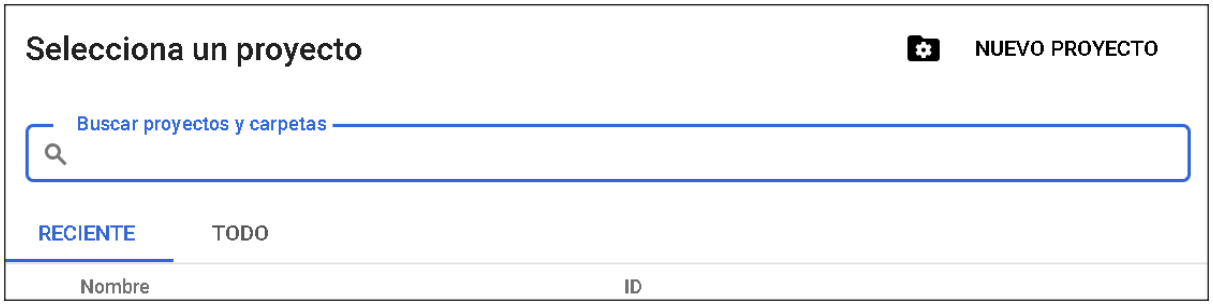
En esta zona crearemos los credenciales de tipo oauth2client. Google como resultado para nuestra aplicación web configura un identificador de cliente y un identificador secreto para nosotros.

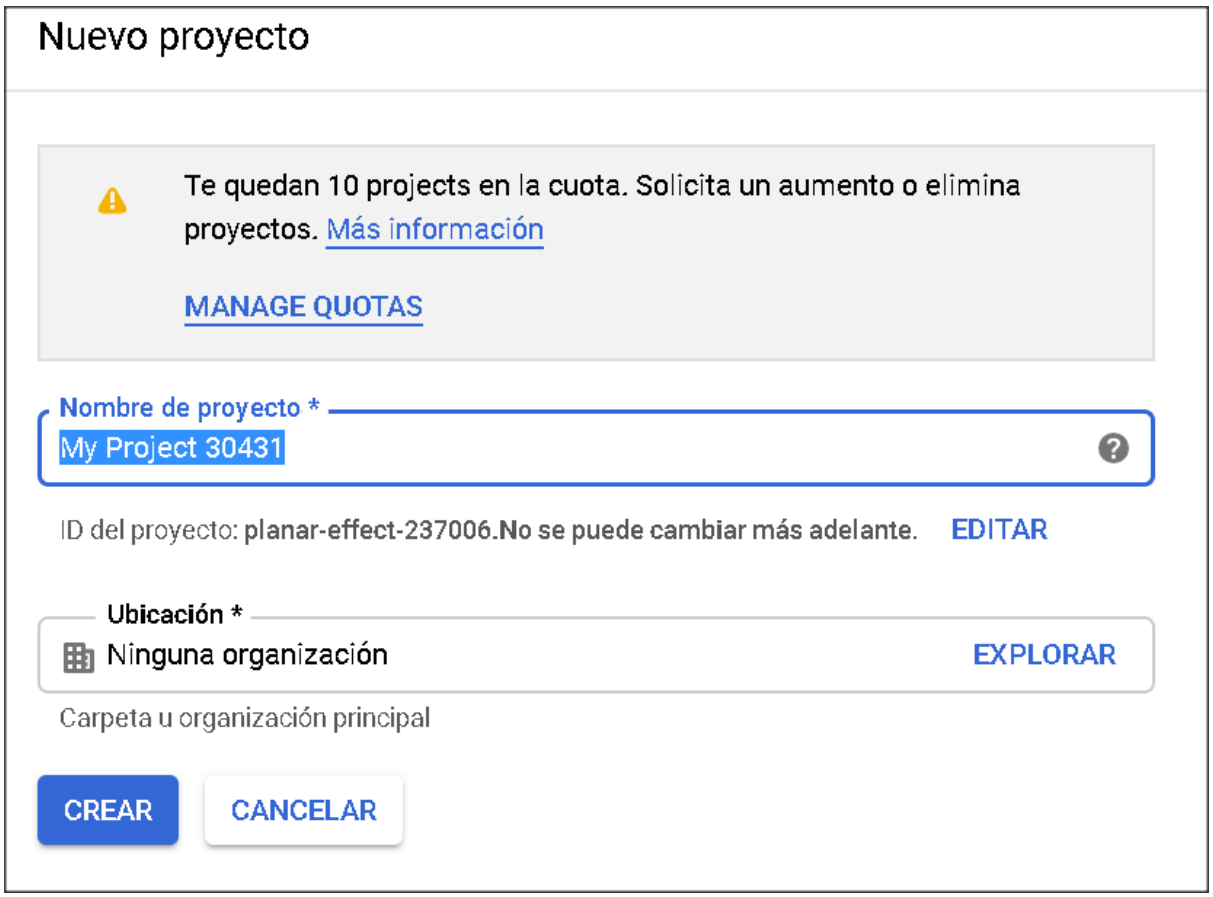
Para ello hay que entrar a la siguiente dirección

https://console.developers.google.com usando una cuenta bajo el paraguas de google.

Primero debemos crear un proyecto para poder configurar los credenciales.

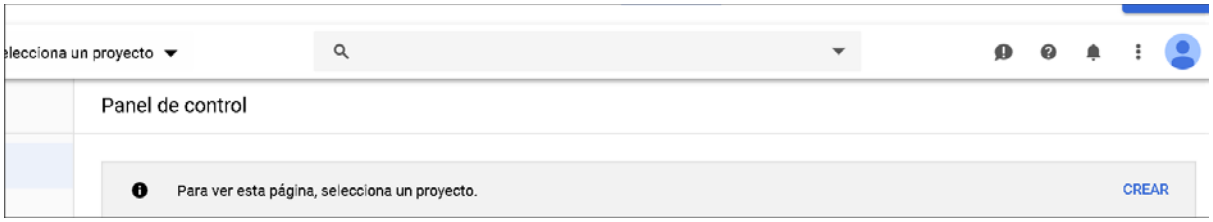
Esta opción nos permitirá crear un nuevo proyecto.



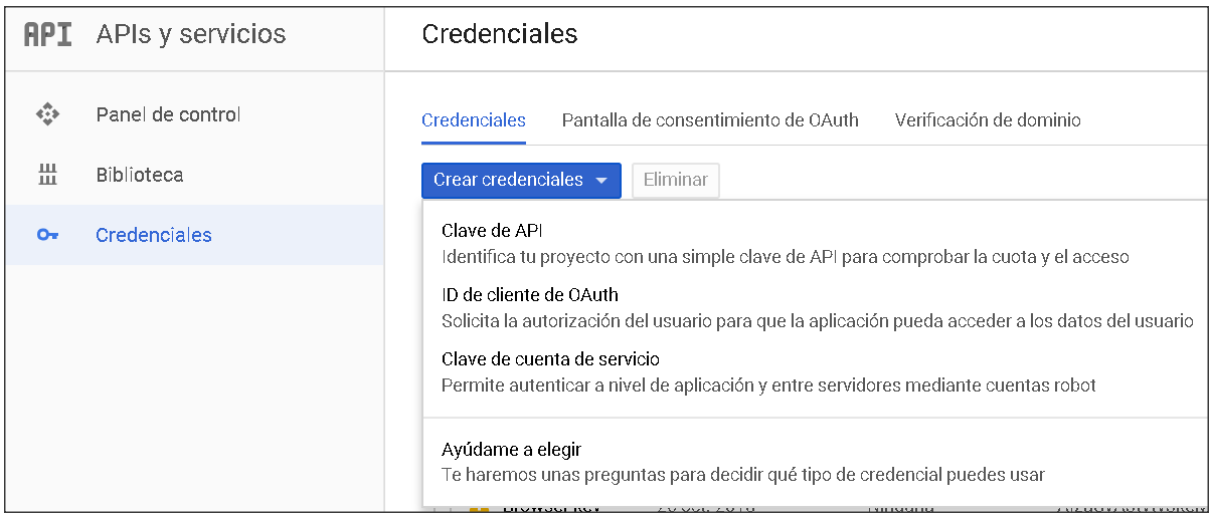


Una vez creado este nuevo proyecto, solo tenemos que seleccionarlo para poder empezar a configurar credenciales Oauth.

Una vez seleccionado el proyecto, vamos por fin a empezar a configurar los credenciales, para ello:



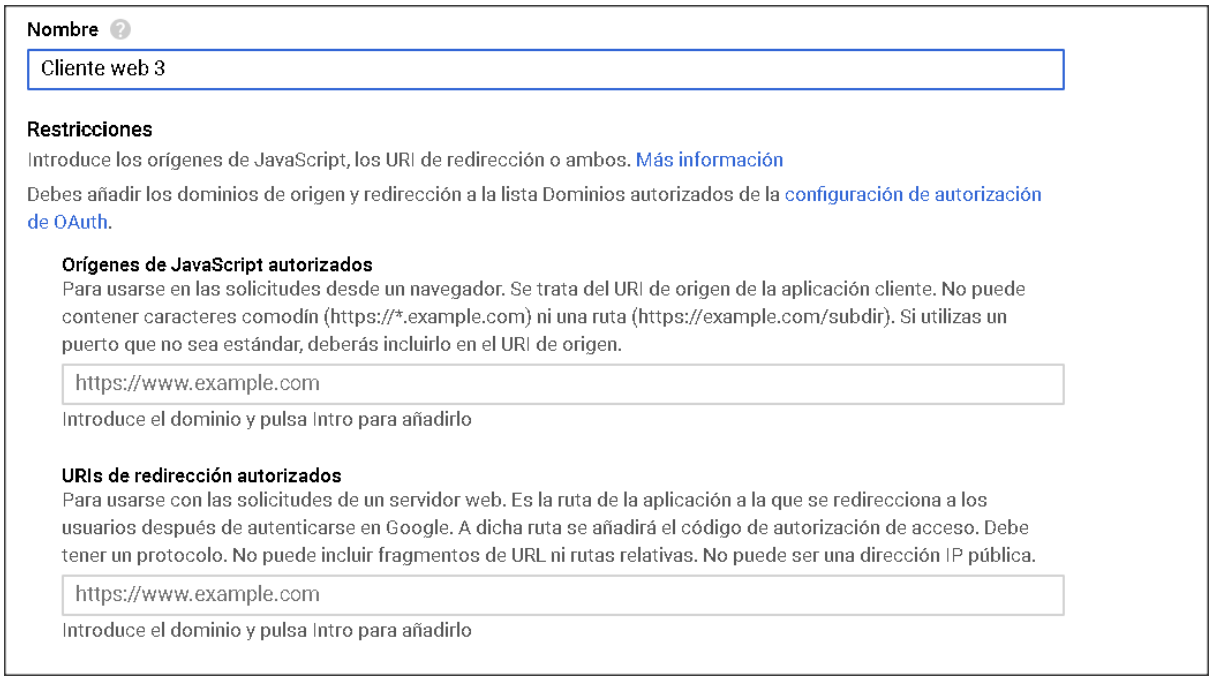
Hacemos clic en la zona de búsqueda y buscamos el término Credenciales.



Una vez dentro de la pantalla, haremos clic en la opción crear credenciales y seleccionaremos la opción de ID de cliente de OAuth.

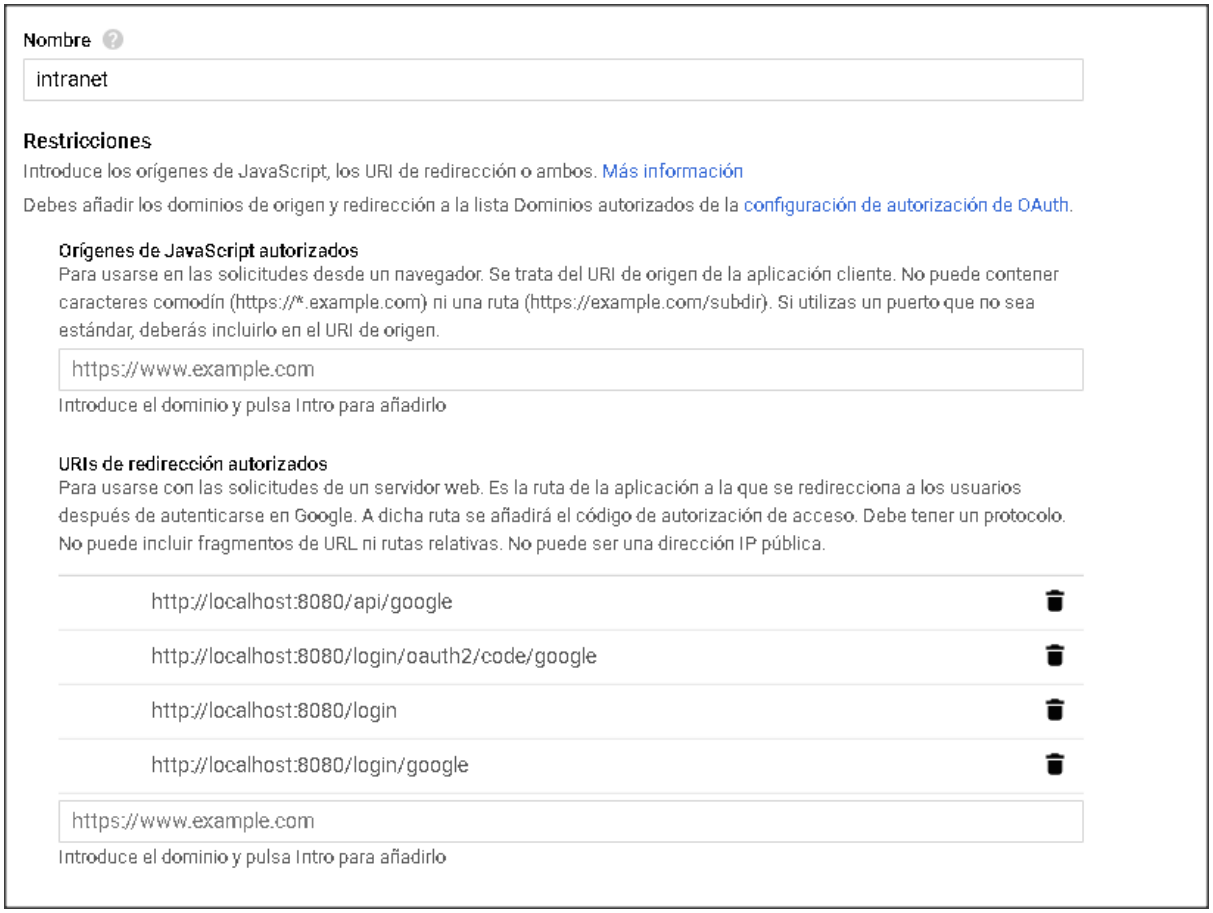
Como vemos google nos ofrece diferentes alternativas dependiendo del tipo de componente o dispositivo que haga uso de su tecnología.

Nos estamos centrando en el desarrollo de una aplicación Web, por tanto escogeremos esa misma opción. Al seleccionar la opción de web, nos aparecerán nuevos parámetros de configuración.



Por el momento bastaría con configurar el nombre, pero en un momento dado, es importantísimo configurar el apartado URI de redirección autorizados.

En nuestro caso y como la aplicación en estos momentos se ejecuta sobre la URI localhost es necesario dejarlo configurado como tal, en un futuro cuando la aplicación esté alojada en un servidor será necesario añadir la nueva configuración correspondiente.



Debemos configurar y autorizar la URI de redirección dentro de la consola de Google. Esta dirección es el lugar donde los usuarios serán redirigidos después de que ellos hayan conseguido loguearse satisfactoriamente con Google. por defecto spring boot configura está URI de redirección cómo */login/google*

Una vez configurado ya podemos ver los detalles de nuestros credenciales otorgados por google.

Ahora ya podemos añadir esta información al fichero aplication.properties de nuestro proyecto.

Si queremos usar un proveedor de autenticación diferente, no configurado en Spring Security, necesitaremos definir completamente la configuración, con información tal como, la URI de autorización la URI del Token y el resto de propiedades que son necesarias.

Spring Security provee unos servicios basados en J2EE muy intuitivos para incorporar en el software de aplicaciones empresariales. Son potentes, flexibles y configurables.

OAuth es un protocolo de autorización abierto que permite el acceso a recursos de un propietario habilitando los servicios HTTP de las aplicaciones cliente.

La especificación Oauth2 define un protocolo de delegación, qué es útil para otorgar decisiones de autorización a lo largo de aplicaciones web a través de la red, asimismo como sobre servicios API.

OAuth es usado en una amplia variedad de aplicaciones, incluído como mecanismo para proveer la autenticación de los usuarios. El entorno Oauth2 permite aplicaciones de terceros obtener acceso limitado a los servicios HTTP. Este acceso a los servicios HTTP está orquestado mediante la aprobación de una organización que actúa de intermediaria entre el propietario de los recursos y el recurso HTTP solicitado, permitiendo así que la aplicación de terceros obtenga acceso en su propio nombre.

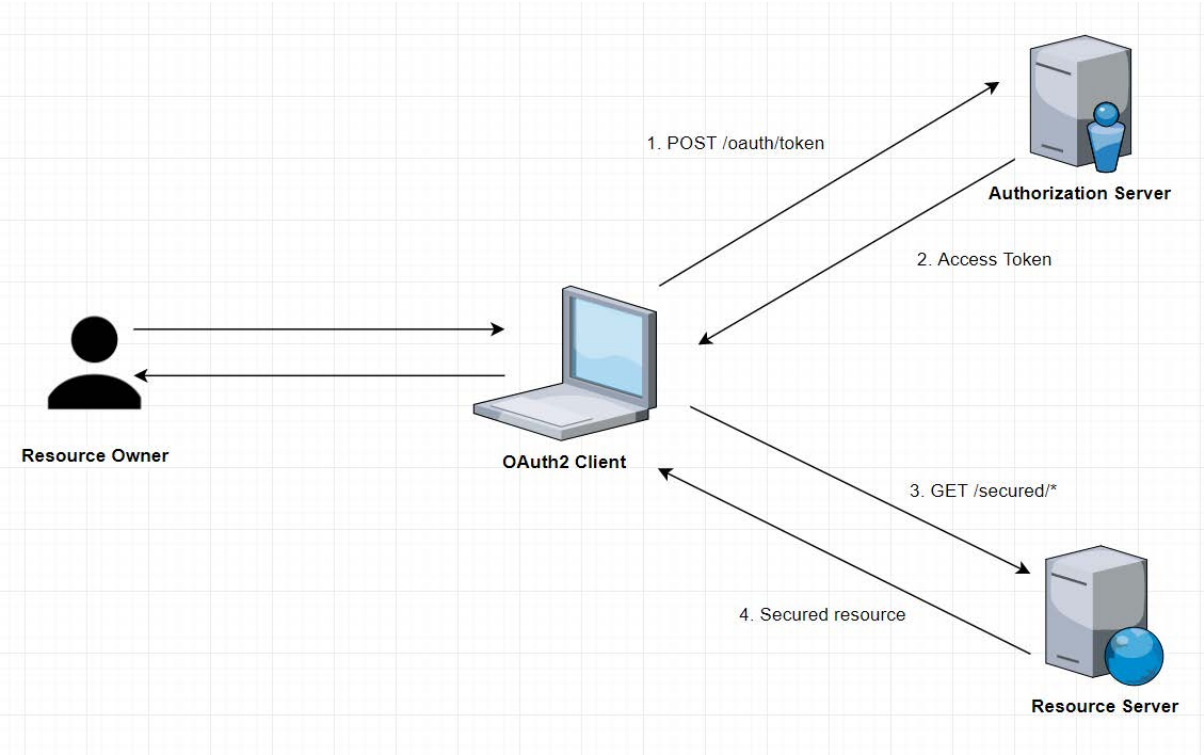
Para ello Oauth utiliza los siguientes roles:

1. Propietario del recurso: el propietario del recurso es una persona, por ejemplo, un usuario final en una aplicación que posee el servicio o la política de seguridad.

2. Servidor del recurso: el servidor de recurso que aloja el recurso o el servicio protegido. Se encarga de servir información que está protegida por el token OAuth.

3. Cliente de aplicación: La aplicación cliente se dedica a solicitar el acceso mediante solicitudes a recursos almacenados en el servidor de recursos. La aplicación cliente también obtiene la autorización desde el propietario de recursos.

4. Servidor de autorización: el servidor de autorización autoriza a la aplicación cliente para acceder a los recursos del propietario de los mismos.



Los token OAuth son implementaciones específicas de cadenas de texto aleatorias generadas por el servidor de autorización.

1. Token de acceso: es enviado con cada solicitud y normalmente es válido durante una hora.
2. Token de refresco es usado para obtener un nuevo token de acceso y nunca se envía con la solicitud. normalmente vive más tiempo que el token de acceso.

Para incorporarlo dentro de la aplicación ha sido necesario añadir al pom.xml las siguientes dependencias

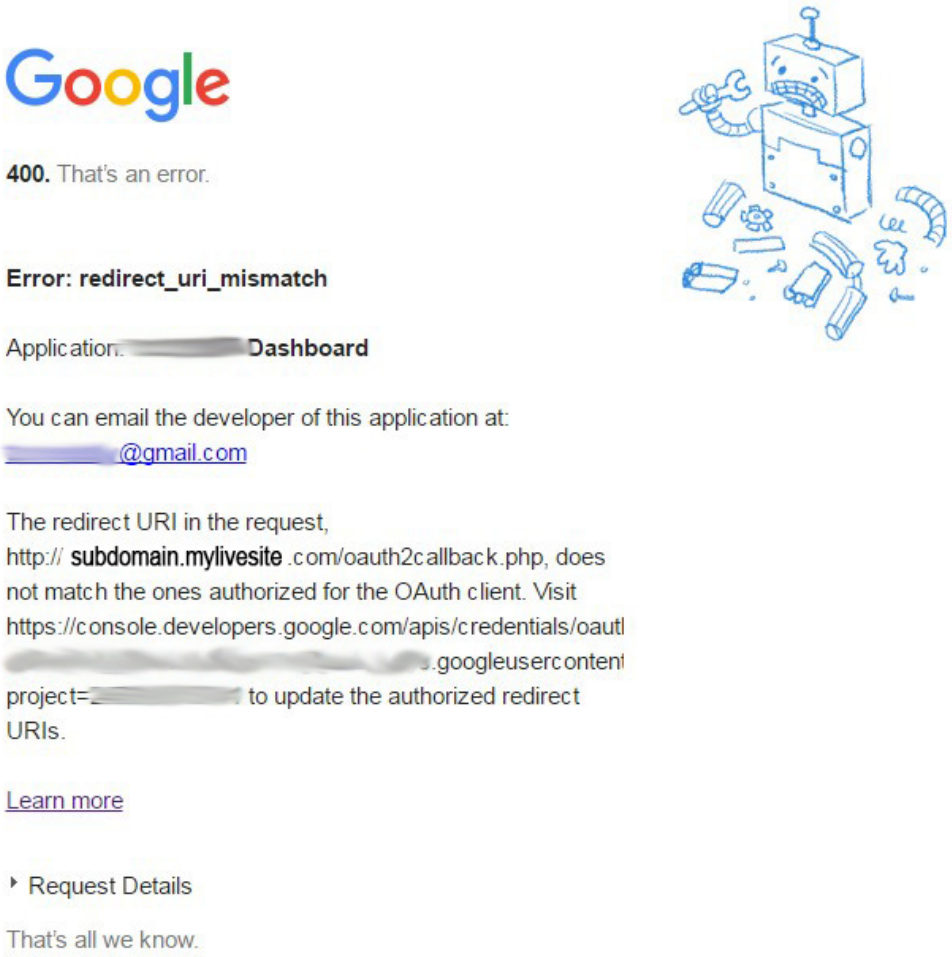
```
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-oauth2-core</artifactId>
    <version>5.1.2.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework.security</groupId>
```

```
            <artifactId>spring-security-oauth2-client</artifactId>
            <version>5.1.2.RELEASE</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.security</groupId>
            <artifactId>spring-security-oauth2-jose</artifactId>
            <version>5.1.2.RELEASE</version>
        </dependency>
```

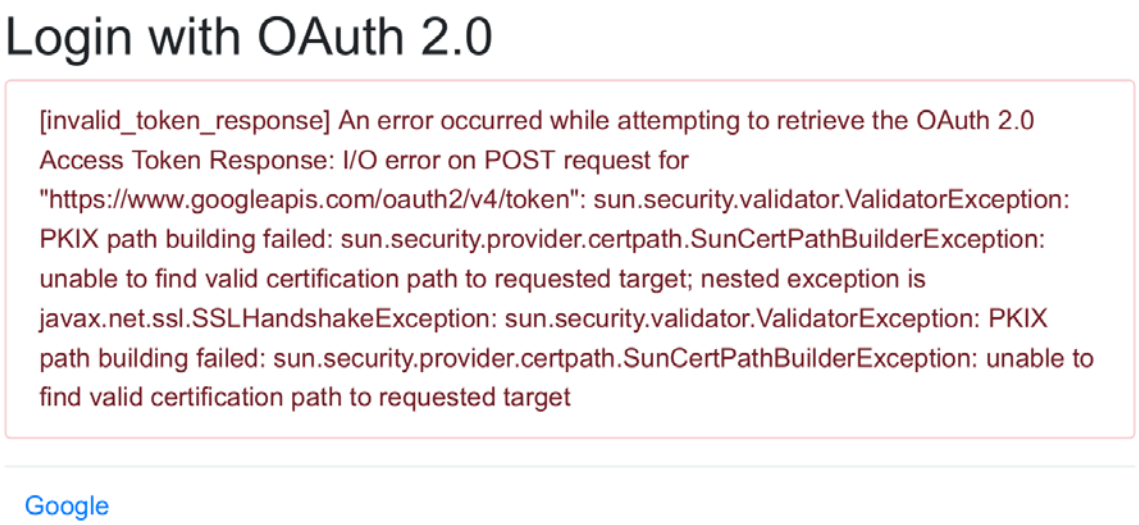
Y además añadir las siguientes propiedades dentro del fichero application.properties

```
spring.security.oauth2.client.registration.google.clientId
spring.security.oauth2.client.registration.google.clientSecret
```

Cabe indicar que cuando se realizaban estas pruebas, nada parecía funcionar, todo parecía incompatible, a pesar de las pruebas se estuvieron varios días que no se conseguía hacer funcionar el SSO, ya que google indicaba un error tras otro.



Cuando ya parecía se quedaba todo bien configurado, siguiendo las líneas indicadas arriba, de repente al probar la aplicación un error críptico….



La información disponible en internet sobre el tema eran infructuosas, todas las pistas conducían por camino que ninguno de ellos permitía dar con la solución adecuada. El error persistía. Ninguno de las soluciones encontradas era la correcta y cada una te llevaba por diferentes caminos.

Finalmente se decidió probar si este mismo problema ocurría con versiones de la máquina virtual java diferentes a la que estaba siendo usada hasta el momento. Usando la JDK8 y la JDK11 el problema desaparecía, es decir solo existía el problema con la versión 11 de Java.

Por lo tanto para el buen desempeño de la aplicación se descarta el uso de la JDK10.

Haciendo memoria recordé este cambio que produjo varios problemas a los desarrolladores debido a este cambio de la jdk 8 a la jdk 10.

- The certificates are really different
    - JDK 10 has 80, while JDK 8 has 151
- JDK 10 has been recently added the `certs`
    - https://dzone.com/articles/openjdk-10-now-includes-root-ca-certificates
    - http://openjdk.java.net/jeps/319

Para evitar el problema se repita y algún desarrollador use la version 10 de java, se va a forzar desde maven, el uso de la versión 11 de java.

```
<plugin>
        <groupId>org.apache.maven.plugins</groupId>
```

```
        <artifactId>maven-compiler-plugin</artifactId>
        <configuration>
            <release>11</release>
        </configuration>
    </plugin>
```

Con esta solución forzamos a que para compilar el proyecto necesitamos JDK11 como mínimo. Debido a este cambio, se aprovecha también para incorporar algunas de las nuevas características que posee esta versión dentro del código de la aplicación. Fundamentalmente se va a hacer uso de la nueva API para trabajar con cadenas de texto y así poder ir poco a poco eliminando dependencias con librerías de terceros como Commons Lang o la clase StringUtils de spring.

Integración del framework de autorización y Apache wicket

Spring Security es un marco de trabajo integrado incluido dentro de el framework Spring Boot, el cual, nos ayuda a gestionar de manera rápida y eficaz todos los aspectos involucrados en la seguridad de nuestra aplicación y de las peticiones HTTP.

Apache wicket por otro lado, es un marco de trabajo mucho menos conocido que otros dentro ecosistema Java, sin embargo, es un marco de trabajo estable que permite una alta escalabilidad y converge una alta capacidad de integración con servicios de terceros.

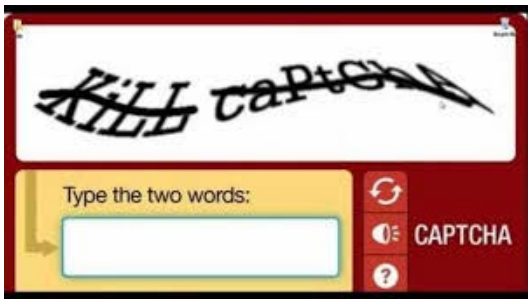
Apache Wicket es un marco de trabajo Java completo, que también se ocupa de controlar y gestionar las terminaciones de las aplicaciones Java, así como, de todas las peticiones web. Por lo tanto, nos encontramos con dos framework que son capaces de realizar la misma tarea. Es sabido que cuando dos actores se ocupan de la misma tarea, sus campos se solapan y ambos dos actores compiten por obtener las competencias sobre ese terreno. En estos casos, el mejor enfoque que se le puede dar al problema, es nombrar a uno de estos actores como el jefe principal o como el orquestador principal de las peticiones, y el otro actor se convierte en un subsidiario del primero.

El segundo actor desconoce que el primero es el encargado de filtrar toda la información que no sea imprescindible. Este segundo actor, secundario, en nuestro caso Apache Wicket solo lo utilizaremos como juez a la hora de decidir qué componentes del propio framework deben ser renderizados, y cuáles no. en cambio usaremos a nuestro actor principal Spring Security para las redirecciones web base de nuestra aplicación, así como para forzar el login mediante OAuth.

Como medida de seguridad adicional en la aplicación, en la acción del cambio de rol por parte del usuario se ha insertado un captcha.

CAPTCHA, (Completely Automated Public Turing test to tell Computers and Humans Apart), Un captcha es un sistema de verificación online que comprueba que un formulario ha sido rellenado por un humano y no por una máquina. Aunque en ocasiones pueden resultar molestos, cumplen su función: evitan que robots, bots o máquinas envíen información a través de los formularios de una web de manera automática e indiscriminada.

Los captcha hoy en día se usan muchísimo por detrás para entrenar sistemas de inteligencia artificial, para ayudar a las Inteligencias Artificiales a reconocer objetos.



Hay captchas de muchas formas y tipos, pero todos se basan en el concepto de ofrecer un reto ante posibles procesos automatizados en envío masivo e indiscriminado de información.

En este caso el captcha no entrena ninguna red neuronal por detrás, sino que simplemente le pide al usuario en una imagen de animales, que seleccione aquellos que reconozca como gatitos.

Es conveniente indicar que los captcha, no son un mecanismo de seguridad y desde luego no son infalibles, simplemente añaden una barrera de complejidad adicional fácil de superar para humanos y difícil de traspasar para una máquina.

Una de las grandes dificultades que han surgido durante el desarrollo de la aplicación es la necesidad de actualizar uno de los componentes base, a la última versión de Apache wicket. Este componente llamado fullcalendar 2 estaba adaptado a la versión 6 de Apache wicket, y debido al uso de tecnologías como Java 11, Spring Security y un largo etcétera, ha sido necesario actualizar la versión de este componente para que fuera compatible con la versión 8 de Apache wicket.

Como resultado del mismo ha sido necesario realizar un fork del proyecto alojado en Github. Para poder abordar una mejora y actualización de la versión de Wicket que se usaba hasta el momento, gracias a que el proyecto fullcalendar 2 usa Maven como piedra angular.

La actualización de versión ha sido relativamente sencilla, sin embargo era necesario probar el componente y como resultado de la actualización, parte del código había dejado de compilar. Ha sido necesario modificar el código base del componente para permitir su integración con la versión 8 de Apache wicket, a continuación expongo el código con la información, las partes que han sido necesario incorporar para hacer compatible el componente, como vemos la mejor opción ha sido incorporar un nuevo commit git solventando el problema..

```
public class FullCalendar extends AbstractFullCalendar implements IBehaviorListener
{
public class FullCalendar extends AbstractFullCalendar implements IRequestListener {
```

```
 abstract class AbstractCallback extends Behavior implements IBehaviorListener {
 abstract class AbstractCallback extends Behavior implements IRequestListener {
```

```
                String     url     =     calendar.urlFor(IBehaviorListener.INTERFACE,
params).toString();
                String url = calendar.urlForListener(params).toString();
```

Una vez realizados estos cambios ha sido necesario comprobar el componente funcionaba correctamente, esto ha sido relativamente sencillo debido a que el componente base, ya incorporaba una serie de test unitarios, que probaban gran parte de la funcionalidad. Debido a ello, con unos pequeños cambios se ha logrado actualizar el componente e integrarlo dentro del stack de tecnologías usado. Ciertamente la mejora incluida ha podido ser probada y verificada como válida en los primeros compases, antes de su utilización dentro del proyecto.

Creación de tareas en segundo plano

Otro de los requisitos de la aplicación, era la ejecución de tareas, con el objetivo de importar datos externos dentro de nuestra aplicación, así como, de generar nueva información dentro de la base de datos de manera aleatoria bajo condicionantes.

Estas tareas, deben de poder ser disparadas por el usuario de nuestra aplicación, siempre y cuando, este usuario tenga el rol asociado correspondiente, uno de los requisitos, era que esta tarea requiere un tiempo de ejecución, el cual, puede variar desde periodos muy cortos de tiempo a periodos extremadamente largos, por tanto, no podemos dejar a nuestro usuario pidiéndole que se mantenga a la espera indefinidamente. Nuestro usuario debe ser capaz de poder interactuar con otras partes de la aplicación mientras la tarea se va ejecutando paulatinamente.

Una de las vías que se estudió fue la de hacer uso de los web workers de JavaScript, sin embargo, su uso añadía una capa complejidad al sistema innecesaria, ya que requería introducir un nuevo lenguaje de programación de manera intrusiva. Con el fin de mantener la aplicación con el mínimo número de lenguajes posible, se optó por encontrar una solución al problema, que fuera compatible con todo el conjunto de tecnologías que se estaba usando.

Como consecuencia de ello se decidió explorar las clases worker que nos ofrece el framework Apache wicket. Estos worker funcionan muy bien de manera independiente cuando el único framework de la aplicación que se está utilizando es Apache wicket. Pero cuando nuestra aplicación basada Apache wicket se encuentra dentro del contexto de Spring su integración no resulta tan intuitiva y hay que buscar mecanismos alternativos para poder realizar la ejecución de la tarea.

Una de las necesidades, es que las tareas deben de ser asíncronas y además tienen que mostrar el progreso de las mismas, para que así el usuario no se sienta frustrado.

Además, se debe incluir una tarea automatizada, que, cada día debe enviar correos a las personas, que al día siguiente tienen la actividad, a modo de recordatorio. En el caso de esta tarea planificada se ha decidido optar por una fórmula parecida a lo que hace linux con las tareas cron.

Las expresiones cron se definen dentro de Spring de la siguiente forma:

```
@Scheduled(cron = "[Seconds] [Minutes] [Hours] [Day of month] [Month] [Day of week] [Year]")
```

Usando esta anotación podemos conseguir que nuestra tarea se almacene dentro del contexto de Spring.

Las partes que componen una expresión cron son:

1. Seconds puede tomar valores 0-59 o los caracteres especiales , - * / .
2. Minutes puede tomar valores 0-59 o los caracteres especiales , - * / .
3. Hours puede tomar valores 0-59 o los caracteres especiales , - * / .
4. Day of month puede tomar valores 1-31 o los caracteres especiales , - * ? / L W C .
5. Month puede tomar valores 1-12, JAN-DEC o los caracteres especiales , - * / .
6. Day of week puede tomar valores 1-7, SUN-SAT o los caracteres especiales , - * ? / L C # .
7. Year puede estar vacío o tomar valores 1970-2099 o los caracteres especiales , - * / .

Antes de continuar, es necesario entender lo que significan los caracteres especiales.

1. * representa todos los valores por ejemplo si se usa en el apartado de Day, la tarea se ejecutará cada día.
2. ? representa un valor no específico y puede ser usado para indicar el primer día del mes o el primer día de la semana
3. - representa un rango de inclusión. Por ejemplo 1-3 en el campo de las horas significa las horas 1, 2 y 3.
4. , representa valores adicionales. Por ejemplo, escribiendo MON,WED,SUN en el dia de la semana significa Lunes, Miércoles, Domingo.
5. / representa incrementos. Por ejemplo 0/15 en el campo de los segundos se activa cada 15 segundos empezando desde 0 (0, 15, 30 y 45).

Debido a que usamos diferentes entornos, configuraremos esta tarea de manera diferente en cada uno de ellos, para el entorno de preproducción y producción esta tarea se ejecutará cada día, en cambio para nuestras pruebas en el entorno de desarrollo y pruebas, la tarea se ejecutará cada menos tiempo, para así comprobar que todo funciona en la forma que debe.

Esta tarea manda un correo electrónico a los usuarios con el siguiente mensaje

"Particiapación en Buenos Dias" Papelera ×

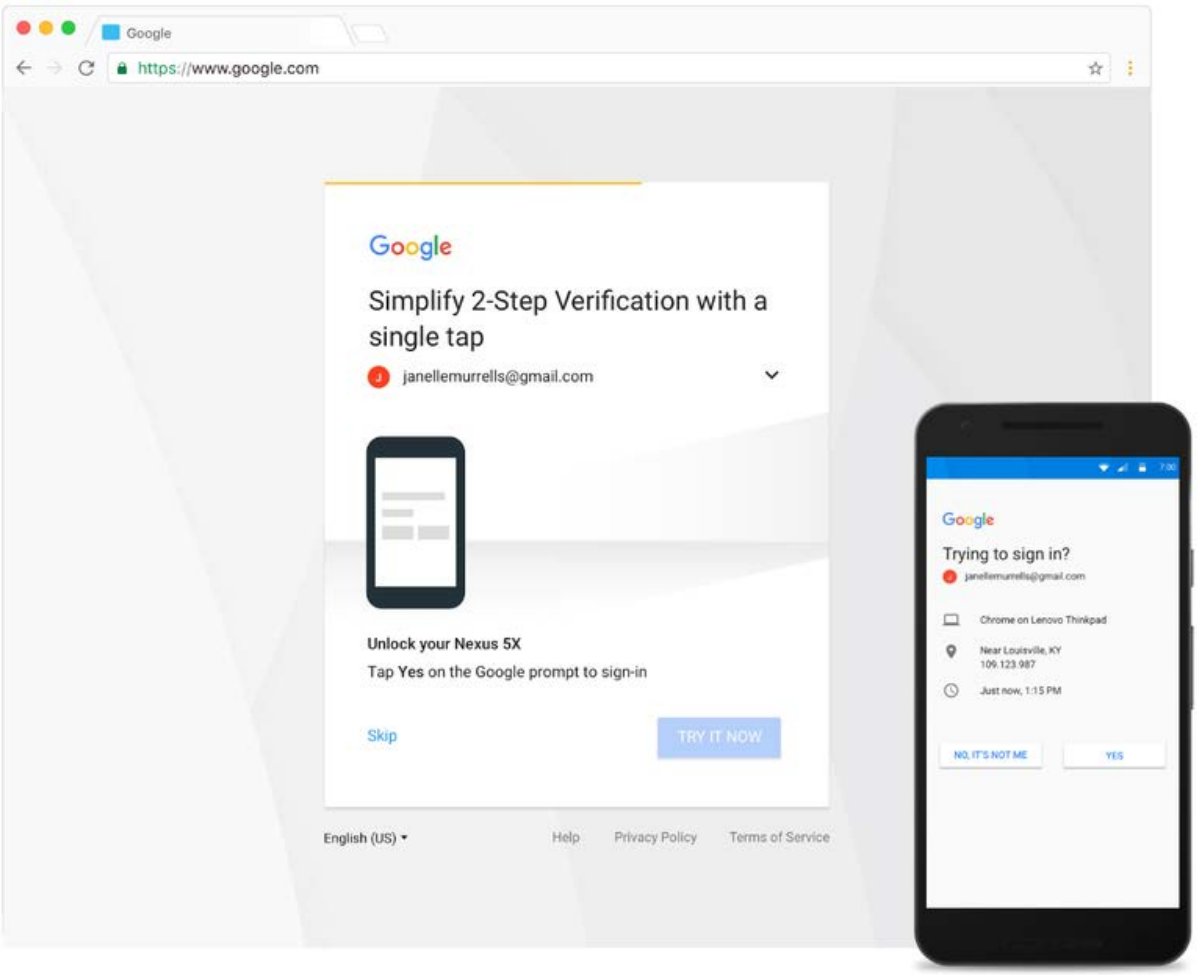
gorka.sanz@salesianos.edu 27 mar. 2019 17:50 (hace 13 días)

para mí

Este mensaje se ha eliminado. Restablecer mensaje

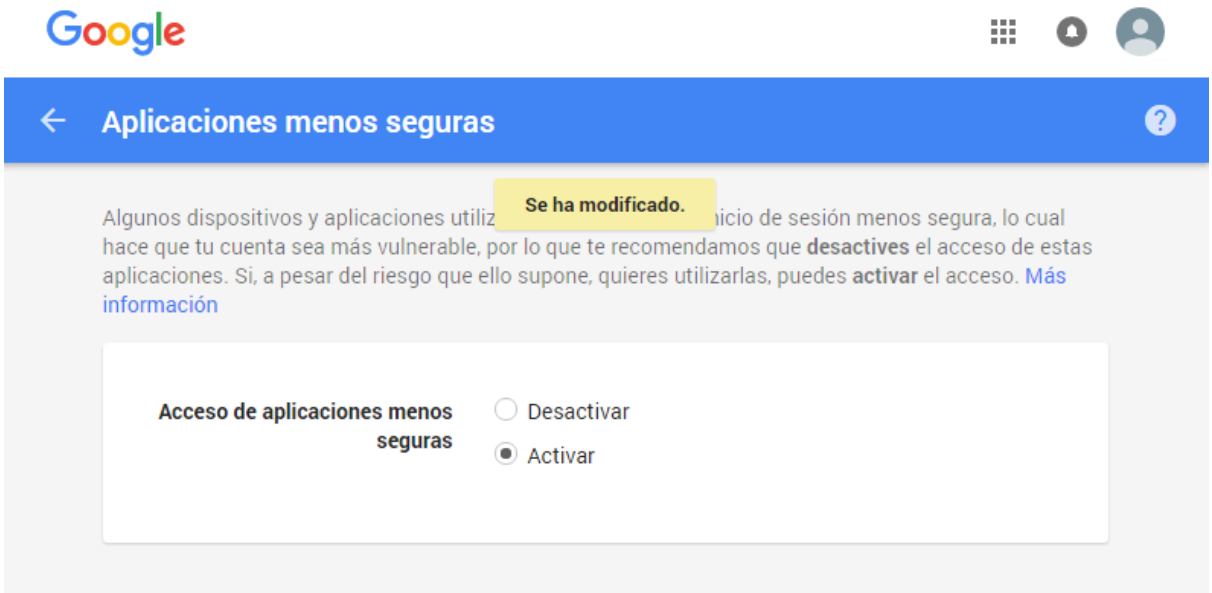
"Me gustaría contar con tu presencia para participar en Buenos Dias el día 03-04-2019 a las 08:00, muchas gracias por tu esfuerzo."

Lograr enviar correos desde una cuenta de gmail, usando la librería java mail, no es una tarea inmediata. En realidad google se ha cerrado en torno a su servicio, y lo que antes usaba un estándar normal de correo electrónico como Pop3, SMTP, IMAP, hoy en día a google no le interesa y obliga a que tengas que hacer uso de su librería API para servicios, como el correo electrónico, el uso de esta librería requiere además, usar la autenticación de google para enviar el correo electrónico, la cosa se complica aún más si en en la cuenta tienes configurado la autenticación en dos pasos con el móvil 2FA.



La única alternativa para enviar correos usando la cuenta corporativa, sin tener que usar la librería de google es activando la acción de [Permitir el acceso a cuentas desde aplicaciones poco seguras]

Se muestra una imagen de la pantalla de configuración donde podemos cambiar esta preferencia.



Activando esta opción, la aplicación ya se vuelve capaz de enviar correos desde la cuenta de correo electrónico, usando una librería bien probada y conocida como javaMail. Este camino es el que emplean aplicaciones Windows phone y Symbian.

Para enviar la información desde un correo electrónico es necesario tener acceso al mismo y por tanto conocer la contraseña de acceso. Esa contraseña jamás debe incluirse en el código de la aplicación y aunque se incluya en un fichero de configuración protegido por el servidor, nadie nos puede asegurar que la seguridad de esa máquina no se encuentre comprometida. El camino más recomendable es incluir la contraseña pero cifrada con algún mecanismo.

La solución investigada usada requiere añadir la siguiente dependencia al proyecto

```
<dependency>
	<groupId>com.github.ulisesbocchio</groupId>
	<artifactId>jasypt-spring-boot-starter</artifactId>
	<version>2.0.0</version>
</dependency>
```

Escribimos lo siguiente en la consola para obtener un cifrado de la información sensible.

```
encrypted-pwd$ java -cp ~/.m2/repository/org/jasypt/jasypt/1.9.2/jasypt-1.9.2.jar
org.jasypt.intf.cli.JasyptPBEStringEncryptionCLI input="contactspassword"
password=supersecretz algorithm=PBEWithMD5AndDES


----ENVIRONMENT-----------------

Runtime: Oracle Corporation Java HotSpot(TM) 64-Bit Server VM 24.45-b08


----ARGUMENTS-------------------

algorithm: PBEWithMD5AndDES

input: contactspassword

password: supersecretz


----OUTPUT----------------------

XcBjfjDDjxeyFBoaEPhG14wEzc6Ja+Xx+hNPrJyQT88=
```

La cadena de texto de salida, será lo que añadiremos al fichero application.properties

```
mail.password=ENC(XcBjfjDDjxeyFBoaEPhG14wEzc6Ja+Xx+hNPrJyQT88=)
```

Por defecto, si no pasamos ningún parámetro algorithm, se usará la encriptación `PBEWithMD5AndDES`, mas que suficiente para el propósito de añadir protección a nuestra aplicación.

En ese momento para arrancar la aplicación, tendremos que pasarle el parámetro usado como clave para cifrar, de alguna forma, esto lo haremos invocando la aplicación, de la siguiente manera.

```
mvn -Djasypt.encryptor.password=supersecretz spring-boot:run
```

Técnicas que permiten un desarrollo ágil y seguro de la aplicación

Más allá de las técnicas bien conocidas para el desarrollo de aplicaciones que nos permite la creación de aplicaciones web seguras y minimizando el número de bugs. entre las cuales se encuentra el uso de código limpio, de pruebas unitarias sobre el código y, de cobertura de código. Con el desarrollo de esta aplicación se ha querido profundizar en el uso de técnicas que permitan el desarrollo de una aplicación escalable. Una aplicación que permita una fácil integración con los diferentes entornos sobre los que se va a desplegar el artefacto.

Para el desarrollo de la aplicación se ha tomado como base la tecnología Maven que permite la construcción de proyectos usando una estrategia de convención sobre configuración. Nos permite aprovecharnos de buenas prácticas ya establecidas dentro de la comunidad del desarrollo y que sabemos funcionan de manera eficaz, esto es algo parecido a la definición formal de lo que suponen los patrones de diseño.

En lugar de restringir estas buenas prácticas al código fuente, se amplía la idea, a la construcción completa de las aplicaciones mediante artefactos. Los artefactos no son otra cosa más que componentes reutilizables y modulares que tienen la capacidad de ser reusados allí donde se requiera siempre y cuando posean interfaces bien definidas y acotadas.

Para el desarrollo sobre el que se asientan las bases de la aplicación, se ha decidido utilizar Spring Boot el cual ahorra bastante trabajo de inicialización a la hora de decidir e incorporar que dependencias encajan dentro del proyecto, por contra, tiene la pega de que las versiones de nuestras librerías deben de funcionar al unísono, lo que conlleva un esfuerzo extra por parte del desarrollador.

Una de las características intrínsecas de la mayoría de aplicaciones web Java es que se basan en la tecnología Servlet y esta tecnología requieren el uso de un servidor de aplicaciones para alojar nuestra aplicación. Conlleva que muchas veces durante el desarrollo (por cada vez que hemos desarrollado una nueva funcionalidad para nuestra aplicación y deseamos probarla debemos empaquetar nuestro proyecto), dedicar un valioso tiempo a desplegar esa nueva versión dentro del contenedor de aplicaciones. Desperdiciando un valioso tiempo que podría ser dedicado a mejorar o incrementar las funcionalidades de nuestro código.

Por ello, se ha decidido integrar el uso del servidor de aplicaciones dentro del propio ciclo de vida de Maven, investigando las posibilidades de Spring boot se descubre que es relativamente sencillo integrar este tipo de aplicaciones y hacer uso de un servidor embebido.

El servidor configurado, ha sido un servidor jetty, el cual, tiene unos unos tiempos de arranque y de parada relativamente cortos. Esta minimización de tiempos permite al desarrollador experimentar cómo se va a sentir el usuario usando la aplicación, reduciendo drásticamente los tiempos para la puesta en marcha.

Por otro lado, Maven permite el uso de perfiles para la construcción de nuestra aplicación: El uso de perfiles ha permitido gestionar todas las posibilidades de los diferentes entornos que nos vamos a encontrar. concretamente se han creado cuatro perfiles que son desarrollo, test, preproducción, y producción. Los definimos en el fichero pom.xml de la siguiente manera:

```
   <profiles>


        <profile>
            <id>dev</id>
            <activation>
                <activeByDefault>true</activeByDefault>
            </activation>
            <properties>
                <env>dev</env>
            </properties>
        </profile>


        <profile>
            <id>test</id>
            <properties>
                <env>test</env>
            </properties>
        </profile>
```

```
        <profile>
            <id>pre</id>
            <properties>
                <env>pre</env>
            </properties>
        </profile>

        <profile>
            <id>prod</id>
            <properties>
                <env>prod</env>
            </properties>
        </profile>

    </profiles>
```

Uno de los requisitos de nuestra aplicación es el de utilizar una base de datos ya construida previamente, esta base de datos es MySQL, la cual, ya contiene una serie de tablas generadas, por lo tanto nos hemos aprovechado de los perfiles de Maven creador para que los perfiles de preproduccion y produccion hagan uso de las librerías necesarias para conectar con MySQL.

Por otro lado los perfiles de desarrollo y de test están configurados para utilizar una base de datos H2. Las bases de datos H2 son bases de datos que pueden residir en memoria o en fichero. Son base de datos muy simples pero a la vez muy potentes que nos permiten desarrollar funcionalidades de nuestra aplicación de manera rápida y minimizando la posibilidad de incompatibilidades con la base de datos del cliente final.

Para el perfil de test se ha utilizado la base de datos H2 establecida en memoria, esta base datos simplemente está viva mientras la aplicación se encuentra operativa y está en

ejecución, en cuanto el proceso de nuestra aplicación se cierra la base de datos también se destruye,

Este tipo de base de datos suelen ser muy usadas integrándose dentro de pruebas de unidad, sobre todo cuando se prueban las funcionalidades de los repositorios, que requieren de una conexión a la base de datos. En estos casos, muchas veces resulta inmanejable configurar una base de datos real para todos los desarrolladores de la aplicación, y se suelen utilizar este tipo de base de datos alojadas en memoria para comprobar el desempeño de los componentes.

Para el perfil de desarrollo también hemos utilizado una base de datos H2 pero en este caso almacena la información de las tablas y de los registros en un fichero, esto nos permite que cuando realizamos cambios a nuestra aplicación, estos cambios sean persistentes y no sean destruidos cuando cerramos el contexto de nuestra aplicación. El uso de estas técnicas permiten incrementar la velocidad de desarrollo y reducir el “time to market” de nuestro producto.

Para configurar la base de datos H2 es necesario configurar el fichero application.properties de la siguiente manera.

```
###
#   Database Settings
###
spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.platform= h2
spring.datasource.username = sa
spring.datasource.password =
spring.datasource.driverClassName = org.h2.Driver
spring.jpa.database-platform= org.hibernate.dialect.H2Dialect
###
#   H2 Settings
###
```

spring.h2.console.enabled=true

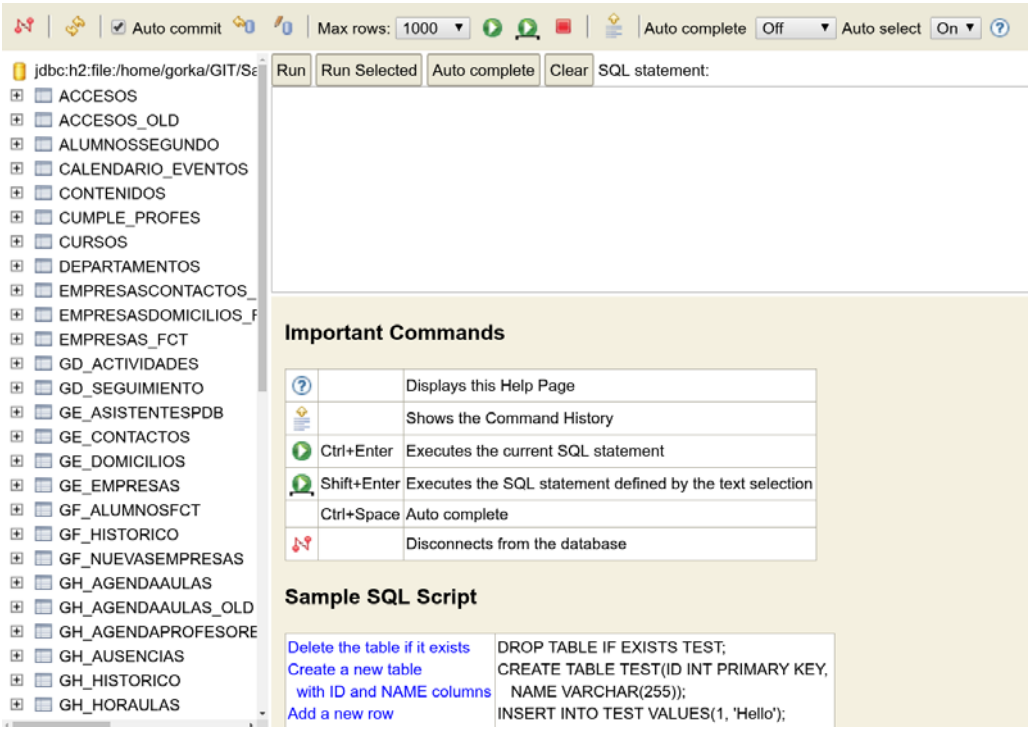
spring.h2.console.path=/console

spring.h2.console.settings.trace=false

spring.h2.console.settings.web-allow-others=false

Con esto configurado así, cuando arrancamos la aplicación, tenemos disponible la siguiente url para acceder al cliente h2, que spring se encarga de configurar por nosotros disponible en *http://localhost:8080/console*

El resultado obtenido sería el siguiente



Spring a partir de su versión 5 nos permite también definir en perfiles de configuración, esta característica. Resulta muy interesante, ya que nos permite también definir esos mismos perfiles que tenemos para Maven, dentro del contexto de nuestra aplicación, de esta forma podemos hacer todo el trabajo de configuración de la persistencia de manera transparente, transversal y sin tener configuraciones repetidas con multitud de sentencias if..else, e incluso switch..case.

Gracias al uso de la inyección de dependencias qué spring nos provee, nuestro código queda mucho más simple y escalable, ya que si en el futuro necesitamos un nuevo perfil, es tan sencillo cómo configurarlo dentro del fichero application.properties de spring.

Como vemos, es mejor mantener la conexión de base de datos en un archivo de propiedades, que permanezca externo a la aplicación y se puede modificar. Sin embargo, Spring Boot, de forma predeterminada, proporciona solo un archivo de propiedades (application.properties). Entonces, ¿cómo vamos a agregar las propiedades?

La solución consiste en crear más archivos de propiedades y agregar el nombre de "perfil" como sufijo en el nombre y configurar Spring Boot para que elija las propiedades adecuadas según el perfil.

Entonces, necesitamos crear tres application.properties:

application-dev.properties

application-test.properties

application-prod.properties

Por supuesto, application.properties permanecerá como el archivo de propiedades maestro, pero si reemplazamos cualquier propiedad en el archivo específico del perfil, esta última ganará prioridad sobre el maestro.

Hemos utilizado el perfil @("Dev") para que el sistema reconozca que este, es el bean que se debe seleccionar cuando configuramos el perfil de la aplicación en DEV. Los otros dos beans no serán creados en absoluto.

Una última configuración es necesaria para hacerle saber al sistema el entorno en el que nos encontramos DEV, TEST o PROD. ¿Pero cómo hacemos esto?

Aquí hay un par de maneras de configurar el perfil activo:

- A la hora de lanzar la aplicación Java.
    - -Dspring.profiles.active=qa - en las propiedades de la VM
- Hacer los siguiente en el fichero application.properties
    - spring.application.profiles=dev

Usaremos la entrada siguiente del fichero application.properties:

```
spring.profiles.active=dev
```

A partir de aquí, Spring Boot sabrá qué perfil elegir.

Solo tenemos que cambiarlo, una vez, en application.properties para que Spring Boot conozca sobre qué entorno se implementa el código, y hará la magia con la configuración.

Una de las herramientas que más tiempo ha permitido ahorrar durante el desarrollo ha sido el uso de las Spring Dev Tools que tiene Spring Boot ya tiene preparadas para ser utilizadas.

Spring Dev Tools permite que cuando se realizan cambios sobre el código fuente con la aplicación iniciada la vez, los cambios en el código fuente provocan un reinicio de la aplicación, pero en lugar de cargar de nuevo la aplicación desde principio solo se cargan aquellas partes que han sido modificadas.

Esta tecnología de Spring, no es la misma que la tecnología usada por JRebel aunque tienen ciertas similitudes, JRebel es un cargador de clases en caliente, y realmente Spring no es tan potente como esto. Lo que hace entre bambalinas es mantener todos los beans cargadas en memoria y solo se re-escribe aquellos que han sido modificados en el código fuente.

Para configurar Spring Dev Tools tan solo tenemos que añadir la dependencia correspondiente a Maven y cuando Spring al arrancar la aplicación, detecta en el contexto de Maven estas dependencias se auto-configura correctamente por sí sola.

```
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
        <version>${springboot}</version>
        <optional>true</optional>
</dependency>
```

Extraer información desde orígenes de terceros

Una de las necesidades del proyecto era poder extraer información de sitios alojados en Internet, para ello era necesario la utilización de una herramienta que nos deje escarbar la información que se encuentra en Internet.

Para ello, se ha decidido explorar el uso de herramientas de scraping web. el scraping es una técnica muy parecida a la que utilizan los robots de los buscadores de Internet, por ejemplo Google, que los usa para indexar toda la información alojada en Internet, incorporando dicha información dentro de sus servidores y de sus bases de datos.

Para este proyecto era necesario extraer toda la información de una web que contiene el calendario de festividades educativas. Esto se conoce como scraping web, es una técnica muy potente y nos permite obtener el DOM de un servidor web para a continuación poder extraer del mismo aquella información que nos interese leer. En pocas palabras, el scraping web es el proceso de extracción de datos de sitios web.

Todo el trabajo se lleva a cabo mediante un código que se denomina "scrapper". Primero, se envía una consulta "GET" a un sitio web específico. Luego, analiza el documento HTML basado en el resultado recibido. Una vez hecho esto, el scrapper busca los datos que necesita dentro del documento y, finalmente, los convierte al formato especificado.

En Python existen potentes librerías que permiten hacer esto y que hoy en día están muy de moda debido al uso de la inteligencia artificial, el Deep learning y el Big Data. Concretamente en Python tenemos librerías como scrapy, pero como nos encontramos dentro del ecosistema de Java se tomado la decisión de qué herramienta utilizar, después de hacer un estudio y valoración de las herramientas existentes.

Para este lenguaje, he decidido decantarme por la opción de JSoup, cómo librería para extraer los datos relevantes de una página HTML.

En la valoración de ventajas e inconvenientes que se encuentran descritos al principio de esta memoria, JSoup se ha impuesto a otras librerías como Jaunt y HtmlUnit.

Se ha decidido extraer la base de información que se encuentra disponible en:

http://www.educaragon.org/calendario/calendario_escolar.asp

Con JSoup podemos usar selectores de Css que nos permiten obtener la información de los días festivos de la comunidad de Aragón

```
<tr>
    <td class="DiaLectivo">3</td>
    <td class="DiaLectivo">4</td>
    <td class="DiaFestivoZaragoza">5</td>
    <td class="DiaFestivo">6</td>
    <td class="DiaFestivoZHT">7</td>
    <td class="DiaFestivo">8</td>
    <td class="DiaFestivo">9</td>
</tr>

```

Cada elemento tr, es decir, cada fila de la tabla representa un més del curso escolar y cada td, representa un día de ese mes que puede ser Lectivo o no, la clave del algoritmo debe ser obtener los días Festivos únicamente, para ello nos aprovecharemos de que los días festivos tienen configurados la propiedad class del código html correspondiente indicando si es:

1. DiaFestivo
2. DiaFestivoZaragoza
3. DiaFestivoZHT

Estas son las tres posibilidades que nos interesan para el objetivo de nuestra aplicación.

Supervisando la configuración, usando las métricas

Por resumirlo brevemente, los actuators de Spring Boot ofrecen funcionalidades listas para en entorno de producción.

Supervisan nuestra aplicación, recopilan métricas, comprenden el estado de nuestros beans, y todo ello sin falta de tener que implementarlo por nuestra cuenta.

Los Actuators se utilizan principalmente para exponer información operacional sobre nuestra aplicación durante la ejecución *(health, metrics, info, dump, env, etc.).*

¿Cómo se instala? Lo primero es agregar la dependencia de Spring Boot Actuator de la siguiente manera.

```
<dependency>
 <groupId>org.springframework.boot</groupId>
 <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
```

Y actualizar nuestras dependencias Maven.

¿Cómo se configura? Para la configuración de los endpoints se debe utilizar el fichero *application.properties* que nos provee Spring Boot con el siguiente formato:

endpoints.[nombre del endpoint].[propiedad a personalizar]

Tenemos tres propiedades para poder personalizar, disponibles:

- **id**: mediante el cual se accederá a este punto final por HTTP.
- **enabled**: si es true es accesible, en caso contrario, no.
- **sensitive**: si es true, necesitará que esté autorizado para mostrar información relevante acerca de HTTP.

Por ejemplo, en el caso de */health* tenemos varias opciones para mostrar la información de salud de nuestra aplicación.

Si queremos mostrar mostrar toda la información de la salud de nuestra aplicación sin falta de estar autenticado, utilizaremos la siguiente instrucción:

```
endpoints.health.sensitive=*

management.security.enabled=false
```

Por el contrario, si queremos mostrar sólo el estado de la salud de nuestra aplicación sin estar autenticado, utilizaremos:

```
endpoints.health.sensitive=false

management.security.enabled=true
```

Por último, si no queremos mostrar nada sin estar autenticado, utilizaremos:

```
endpoints.health.sensitive=true

management.security.enabled=true
```

Por defecto, la información que nos aparece en el caso del endpoint */health* es:

```
{
"status" : "UP"
 }
```

Como vemos, esta información podría ser suficiente para saber el estado de nuestro sistema, en este caso que está levantado.

Pero podemos ver más información, en función de cómo configuremos las properties de los actuators, como por ejemplo:

```
{
 "status" : "DOWN",
 "myHealthCheck" : {
 "status" : "DOWN",
 "Error Code" : 1
 },
 "diskSpace" : {
 "status" : "UP",
 "free" : 209047318528,
 "threshold" : 10485760
 }
}
```


En el caso de ***/info*** podemos customizar la información que nos muestra de la siguiente manera:

```
info.app.name=Spring Application
info.app.description=Trabajo final de Máster
info.app.version=1.0.0
```

Y la respuesta del endpoint sería:


```
{
 "app" : {
 "version" : "1.0.0",
 "description" : "Trabajo final de Máster",
 "name" : "Spring Application"
 }
```

```
}
```

Para las métricas utilizaremos **/metrics** que publica información sobre el sistema operativo, la JVM, y más tipos de métricas.

En la documentación oficial de Spring Boot existen muchos endpoints para los actuators que nos pueden ser de ayuda para nuestros entornos. Por resumir, algunos de los más importantes son **/health**, **/info, /metrics**, **/trace.**

Se muestra la información completa a continuación:

ID	Descripción
`auditevents`	Expone la información de eventos de auditoría para la aplicación actual.
`beans`	Muestra una lista completa de todos los spring beans en su aplicación.
`caches`	Expone cachés disponibles.
`conditions`	Muestra las condiciones que se evaluaron en las clases de configuración y configuración automática y los motivos por los que coincidieron o no.
`configprops`	Muestra una lista de todas las `@ConfigurationProperties`.
`env`	Muestra las propiedades de Spring `ConfigurableEnvironment`.
`flyway`	Muestra cualquier migración de base de datos flyway que haya sido aplicada.

`health`	Muestra información acerca de la salud de nuestra aplicación. Un simple “estado” si estamos sin autenticar o información mucho más detallada si estamos autenticados en la aplicación.
`info`	Muestra información arbitraria de nuestra aplicación.
`integrationgraph`	Muestra el gráfico de integración de Spring.
`loggers`	Muestra y permite modificar la configuración de los loggers en la aplicación.
`liquibase`	Muestra cualquier integración con migraciones de base de datos Liquidbase que hayan sido aplicadas.
`metrics`	Muestra la información de métricas de nuestra aplicación.
`mappings`	Muestra una lista completa de todos las rutas `@RequestMapping`.
`scheduledtasks`	Muestra todas las tareas programadas de la aplicación.
`sessions`	Permite la recuperación y eliminación de sesiones de usuario de un almacén de sesiones respaldado por Spring Session. No está disponible cuando se usa el soporte de Spring Session para aplicaciones web reactivas.
`shutdown`	Permite que la aplicación se cierre con elegancia.
`threaddump`	Realiza un thread dump.

`trace`	Muestra información de seguimiento (por defecto las últimas peticiones HTTP).

Valoración del impacto de las pruebas y aspectos de seguridad que han sido implementados.

Cuestionario de impresiones del usuario

Se han realizado pruebas de verificación y validación con tres usuarios, recogiendo sus impresiones mediante un formulario de satisfacción personal cuyas preguntas se encuentran en los anexos de esta memoria, así como los resultados recogidos, los cuales se comentan a continuación.

El feedback general a raíz de los resultados ha sido positivo. En líneas generales las respuestas del formulario no indican carencias graves de la aplicación. El Usuario 3 ha sido el más crítico en sus impresiones, si bien no pone en relevancia problemas graves, sí hace alusión de que la aplicación tiene margen de mejora.

Como estos usuarios han probado el producto final ha sido imposible obtener sus impresiones mediante preguntas durante la realización y consecución de los objetivos 2 y 4 de esta memoria. Por otro lado se han intentado realizar cuestiones que permitieran obtener de forma numérica su grado de satisfacción con la implementación e incorporación del SSO.

Pruebas de funcionamiento

Para comprobar las funcionalidades del producto, se han desarrollado múltiples test, para la comprobación y verificación, si bien, los test no cubren el 100% de las funcionalidades si que establecen un punto a seguir como lanzadera desde el que añadir nuevas pruebas de seguridad.

El impacto de añadir los test ha sido notorio, tanto spring como wicket habilitan herramientas, clases y métodos para probar el código de una manera muy sencilla y legible.

Para ver el resultado de los test ejecutamos desde la consola y el terminal el siguiente comando.

```
>mvn test
[INFO] Scanning for projects...
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
```

[INFO] Running es.******.schedule.ScheduledJobTest

[WARNING] Tests run: 1, Failures: 0, Errors: 0, Skipped: 1, Time elapsed: 0.01 s - in es.salesianos.edu.schedule.ScheduledJobTest

[INFO] Running es.******.edu.utils.SchoolarCalendarScrapperTest

[ERROR] Tests run: 1, Failures: 1, Errors: 0, Skipped: 0, Time elapsed: 1.191 s <<< FAILURE! - in es.salesianos.edu.utils.SchoolarCalendarScrapperTest

[ERROR] test Time elapsed: 0.721 s <<< FAILURE!

java.lang.AssertionError: expected:<holiday> but was:<null>

at es.********.utils.SchoolarCalendarScrapperTest.test(SchoolarCalendarScrapperTest.java:32)

[INFO]

[INFO] Results:

[INFO]

[ERROR] Failures:

[ERROR] SchoolarCalendarScrapperTest.test:32 expected:<holiday> but was:<null>

[INFO]

[ERROR] Tests run: 1, Failures: 1, Errors: 0, Skipped: 0

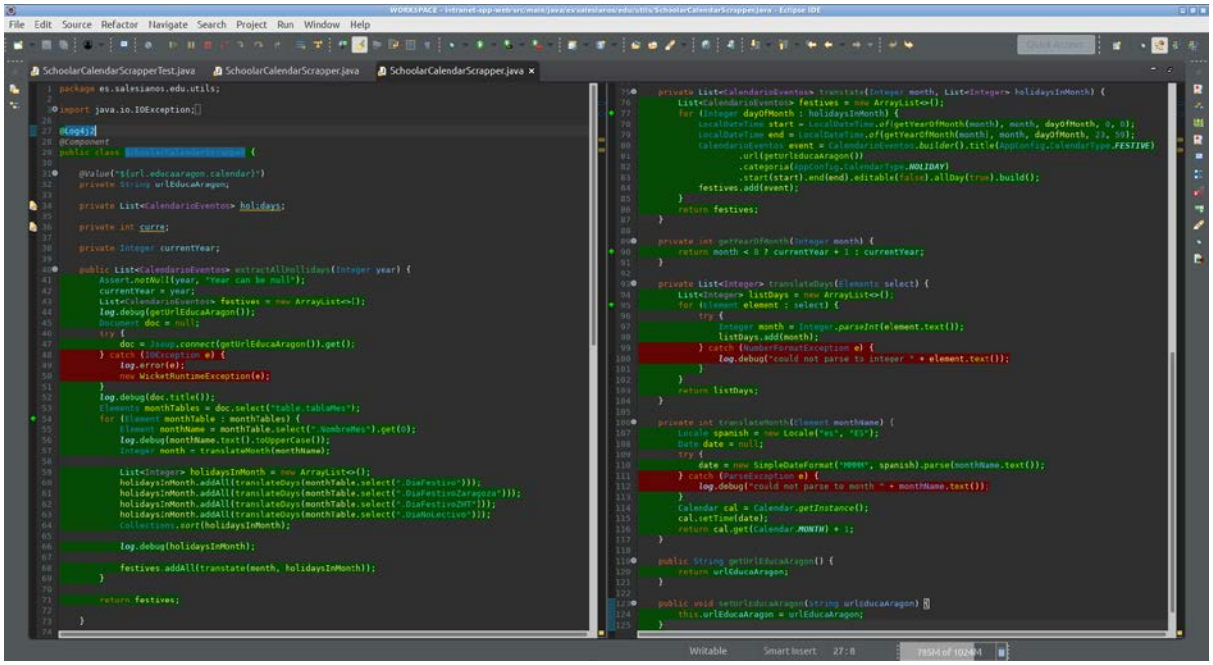
[INFO]

Se puede ver el ejemplo de un test fallido en uno de los módulos que nos advierte de un problema en uno de los test. Nos indica que falla mediante aserciones que si no se cumplen rompen el test.

Este tipo de indicaciones viene bien para reducir la deuda técnica a largo plazo por un mal diseño de nuestra aplicación, además previenen y mitigan los posibles los daños colaterales durante el desarrollo de nuevas funcionalidades.

En el anexo podemos ver la salida del número de test totales realizados en la aplicación, que si bien por motivos de tiempo no han permitido obtener una cobertura 100% del código, si que se han realizado sobre partes críticas de la aplicación que forman parte del núcleo importante.

A continuación se muestra una captura con un ejemplo de cobertura de los test de unidad sobre una de las clases del código.



Como se puede apreciar en esta muestra, los test realizados no cubren el 100% del código, los casos de excepción se quedan sin cubrir por las pruebas, teóricamente se deberían realizar los test necesarios para cubrir también estas partes.

Aún así siempre es mejor tener algo que no tener nada, y las pruebas son algo necesario para conseguir que nuestro proyecto rejuvenezca y conserve la salud por mucho tiempo.

Conclusiones y valoración personal

En esta sección pretendo aportar una valoración y una justificación de lo que este trabajo final de Máster ha supuesto para mí. Por una lado antes de realizar este trabajo ya tenía unos conocimientos previos sólidos sobre cómo realizar aplicaciones debido a mi bagaje laboral previo, conocimientos de lo que aprovecharme para explorar nuevas formas de establecer buenas prácticas a la hora de realizar desarrollos. Al comenzar el proyecto no sabía si las piezas que iba a utilizar iban a encajar o a mitad de camino iba a tener que renunciar a mis ideas y hacer un trabajo final más sencillo, más simple.

Para mí ha sido un trabajo muy personal en el que he podido aprender nuevas cosas que poner en práctica en el ámbito laboral, que me han permitido profundizar mucho más en tecnologías muy demandadas,

No se si el día de mañana las tecnologías aqui descritas quedarán obsoletas, o habrán salido al mercado herramientas mejores que sustituyan a las actuales, lo que sí que sé, es que lo nuevo que salga será fácilmente acoplable siempre y cuando se respete un diseño de arquitectura de la aplicación solido.

Dos grandes obras que ya conocía antes de trabajar en este proyecto son Clean Code y Clean Architecture, ambas, obras de Robert C. Martin, mucha de la filosofía de sus lecturas, las he puesto en práctica en este diseño aunque a lo largo del documento, aunque no haya hecho mención a ello. Para mí, lo que explica y comunican esos libros, son ideas que llevo interiorizadas tan profundo que para mí resultan naturales, son caminos que se abren ante mí que decido tomar sin necesidad de manifestar a los demás que son esas ideas las que pongo siempre en práctica en mis diseños.

Desde luego, todo código es mejorable, y el mío aún más, pero sin duda la base del código realizado sirve de lanzadera y de patrón a seguir para añadir nuevas funcionalidades y crear o añadir nuevos elementos, como pueden ser los test, que ofrecen una confiabilidad extra y una seguridad que aporta solidez al diseño.

Sin duda este proyecto seguiŕá creciendo, añadiendo nuevas metas que alcanzar y recorriendo el camino para llegar a ellas, mirando atrás en el tiempo, al comienzo de este proyecto dudaba de si todas las piezas que deseaba incorporar lograrían encajar, no encontraba ninguna información en la red relativa a lo que quería realizar, y por momentos dudé de que, si nadie lo había documentado, era, porque nadie lo había logrado. Ha costado mucho tiempo, ha costado muchos tutoriales, guías y artículos, sueltos sobre los que

investigar, profundizar y encajar, pero finalmente todas las piezas iban encajando una tras otra como un enorme rompecabezas.

Sin más agradecer, la posibilidad de haber realizado este Máster el cuáĺ me ha servido como un entrenador para no oxidarme y así tener un reto para actualizar mis conocimientos.

Futuras líneas de desarrollo e investigación

Durante el desarrollo de este proyecto hemos podido introducir diferentes técnicas que nos permiten integrar múltiples tecnologías con el objetivo de que trabajen todas al unísono. Se han explorado unas cuantas técnicas de integración entre tecnologías. El proyecto ha permitido conocer en mayor profundidad cómo dichas tecnologías trabajan internamente llevándolas hacia el límite, hacia puntos donde en principio parece que no están preparadas para trabajar juntas, sin embargo tras un exhaustivo análisis y estudio se ha logrado conseguir la integración de los múltiples servicios de una forma transparente.

Como futuras líneas de investigación se recomienda explorar la integración de todos estos servicios con nuevas funcionalidades de Google, como la capacidad de establecer órdenes mediante el uso del asistente de voz.

Además, como línea de investigación se propone añadir nuevas medidas de seguridad que garanticen desarrollar una aplicación más segura y menos proclive a vulnerabilidades de seguridad y a brechas que corrompan la integridad de los datos.

Nos encontramos en una época, donde cada vez más, los problemas de desarrollo, de escalabilidad y de seguridad están cada vez más presentes, por tanto es necesario establecer unas políticas y unas líneas de operación que permitan solventar/minimizar desde el principio todos los futuros problemas que puedan aparecer.

Bibliografía

Para la elaboración de la memoria, así como para la elaboración de la aplicación se han usado los siguientes recursos como guías.

- https://github.com/programmingexperience/OnionArchitecture
- Evans, E., Domain-Driven Design - Tackling Complexity in the Heart of Software, 2004, Addison-Wesley.
- https://spring.io/docs
- https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/
- https://wicketstuff.org/
- https://wicket.apache.org/

ANEXOS

Propiedades de configuración de los diferentes entornos

DEVELOPMENT

Es el entorno usado de manera preferente para desarrollar las funcionalidades y características de la aplicación de forma cómoda sin toda la configuración sobredimensionada que suele requerir de servicios externos, los cuales en ocasiones, a pesar de encontrarse dentro de la intranet del sistema, pueden encontrarse temporalmente no disponibles.

```
###
#   Database Settings
###
spring.datasource.url=jdbc:h2:file:./src/main/resources/dev
spring.datasource.platform=h2
spring.datasource.username = sa
spring.datasource.password =
spring.datasource.driverClassName = org.h2.Driver
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
logging.config=classpath:log4j2-dev.xml

###
#   H2 Settings
###
spring.h2.console.enabled=true
spring.h2.console.path=/console
spring.h2.console.settings.trace=false
spring.h2.console.settings.web-allow-others=false
```

```
###
#   Hibernate Settings
###
spring.jpa.hibernate.ddl-auto = create-drop
spring.jpa.properties.hibernate.show_sql=true
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.use_sql_comments=false
spring.jpa.properties.hibernate.format_sql=false


###
#   Spring actuator Settings
###
management.endpoints.web.exposure.include=*
logging.file=app.log
logging.path=logs/
```

TEST

Es el entorno sobre el que se ejecutan los test de junit fundamentalmente ya que la base de datos usada reside en memoria y se destruye automáticamente al final de cada uso.

```
###
#   Database Settings
###
spring.datasource.url=jdbc:h2:mem:testdb;DB_CLOSE_DELAY=-1;DB_CLOSE_ON_EXIT=false
spring.datasource.platform=h2
spring.datasource.username = sa
spring.datasource.password =
spring.datasource.driverClassName = org.h2.Driver
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
devtools.restart.additional-exclude: /**/*.js, /**/*.css, /**/*.html, /**/*.htm
logging.config=classpath:log4j2-dev.xml

###
#   H2 Settings
###
spring.h2.console.enabled=true
spring.h2.console.path=/console
spring.h2.console.settings.trace=false
spring.h2.console.settings.web-allow-others=false
```

```
###
#   Hibernate Settings
###
spring.jpa.properties.hibernate.show_sql=false
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.use_sql_comments=false
spring.jpa.properties.hibernate.format_sql=false
management.endpoints.web.exposure.include=*
```

PRE

Este entorno es una copia exacta del entorno final del cliente, es usado para probar el estado final de la aplicación, una de las cosas que suele variar de este entorno, al entorno final suele ser el nivel de log.

```
###
#   Database Settings
###
spring.datasource.url=jdbc:mysql://e*****2.informatica**********.org.es:3306/w*****2
spring.datasource.username=
spring.datasource.platform=mysql
spring.datasource.password=
spring.datasource.driver-class-name = com.mysql.jdbc.Driver
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=validate
logging.config=classpath:log4j2-dev.xml

###
#   Sptring actuator Settings
###
logging.file=salesianos.log
logging.path=/var/log/app/
```

PRO

El entorno final del cliente, se detalla la configuración, del mismo, si bien por temas de privacidad se a ocultado, la información sensible.

```
###
#   Wicket  Settings
###
wicket.core.settings.general.configuration-type=deployment


###
#   Database Settings
###
spring.datasource.url=jdbc:mysql://w*****2.informatica***************.org.es:3306/w*****2
spring.datasource.username=
spring.datasource.platform=mysql
spring.datasource.password=
spring.datasource.driver-class-name = com.mysql.jdbc.Driver
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=validate
logging.config=classpath:log4j2-pro.xml
# The SQL dialect makes Hibernate generate better SQL for the chosen database
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5InnoDBDialect
```

Cuestionario con respuestas de la satisfacción del usuario.

Puntúe de con una nota numérica de 1 a 5, su grado de satisfacción con las siguientes cuestiones, siendo 1(nada de acuerdo),2(un poco de acuerdo), 3 (ni de acuerdo, ni en desacuerdo), 4 (bastante de acuerdo), 5(totalmente de acuerdo).

1. Usar una única contraseña, la de mi cuenta google, me da confianza en la aplicación y hace que no tenga que estar tan pendiente de acordarme de una nueva contraseña.
2. La funcionalidad de usar un captcha en la aplicación refuerza mi sensación de seguridad en la aplicación.
3. Se echa de menos una posibilidad de restaurar la contraseña en caso de olvido.
4. La ejecución de tareas en segundo plano resulta agradable para mi, como usuario, cuando me encuentro con tareas en las que debo esperar más de la cuenta.
5. Es agradable ejecutar una tarea, y ver cómo al navegar entre páginas esta tarea no se pierde, ni se destruye, sino que sigue ejecutándose.
6. El feedback del porcentaje de completitud de las tarea en segundo plano es adecuado a mis necesidades.
7. La posibilidad de que la aplicación me recuerde mediante un correo electrónico mis futuras acciones lo veo como algo interesante y estoy satisfecho con el resultado logrado.
8. Sería interesante que además/en lugar de recordatorios mediante correo electrónico, que la aplicación anotara un nuevo evento dentro del calendario personal.
9. La aplicación en líneas generales parece confiable y pienso que mi información estará a salvo de posibles atacantes.
10. La aplicación cumple mis expectativas, y voy a usarla en mi día a día.

	Valoraciones (de 1 a 5)		
Cuestiones	**Usuario 1**	**Usuario 2**	**Usuario 3**
1	5	5	4
2	5	5	5
3	3	4	4
4	5	5	5
5	5	5	5
6	5	5	4
7	5	5	5
8	4	4	4
9	5	5	4
10	5	5	5

Información de salida sobre los test ejecutados

>mvn test

[INFO] Scanning for projects...

[INFO] ---

[INFO] T E S T S

[INFO] ---

[INFO] ---

[INFO]

[INFO] Tests run: 31,Passed: 30 Failures: 0, Errors: 0, Skipped: 1

[INFO]

[INFO] ---