**GRADO EN INGENIERÍA INFORMÁTICA**

Trabajo de Fin de Grado

DESARROLLO DE UNA APLICACIÓN WEB PARA LA DISTRIBUCION DE ESPACIOS

Autor: Alejandro Campos Calvo

Tutor: Nombre y Apellidos del tutor

|  |  |
| --- | --- |
| **UNIVERSIDAD DE LEÓN**  **Escuela de Ingenierías Industrial, Informática y Aeroespacial**  **GRADO EN INGENIERÍA INFORMÁTICA**  **Trabajo de Fin de Grado** | |
| **ALUMNO:** Alejandro Campos Calvo | |
| **TUTOR:** | |
| **TÍTULO:** Desarrollo de una aplicación web para la distribución de espacios | |
| **TITLE:** | |
| **CONVOCATORIA:** Julio, 2019 | |
| **RESUMEN:**  En este trabajo se realiza el desarrollo de una aplicación web para la distribución, visualización y reserva de espacios. El principal objetivo es automatizar el proceso de reserva de espacios, a través de una interfaz de usuario intuitiva y eficiente, que permita de un solo vistazo comprobar que espacios están liberados para su posible reserva en una fecha determinada.  Se estudiará la posibilidad de utilizar la aplicación para facilitar la distribución de aulas de la facultad de ingenierías de la Universidad de León, por lo que la aplicación contará con una comunicación bidireccional con la API de Google Calendar, dando lugar a la actualización en tiempo real de cada uno de los calendarios correspondientes a las asignaturas de los grados y másteres de esta facultad.  Las reservas realizadas a través de la aplicación se verán reflejadas a su vez en cada uno de los calendarios de Google de la asignatura correspondiente, asegurando que no existirán solapamientos de espacios entre los horarios de las diferentes asignaturas de la facultad. | |
| **ABSTRACT:** | |
| **Palabras clave:** Aplicación Web, Distribución de espacios, Google Calendar, API. | |
| **Firma del alumno:** | **VºBº Tutor/es:** |

Í Índice de contenidos

Í Índice de contenidos I

Índice de figuras II

Índice de tablas III

Glosario de términos IV

Introducción 1

Planteamiento del problema 1

Objetivos 1

Metodología 1

Estructura del trabajo 1

1 Estudio del problema 2

1.1 El contexto del problema 2

1.2 El estado de la cuestión 2

1.3 La definición del problema 2

2 Gestión de proyecto software 3

2.1 Alcance del proyecto 3

2.2 Plan de trabajo 3

2.3 Gestión de recursos 3

2.4 Gestión de riesgos 3

3 Solución 5

3.1 Descripción de la solución 5

3.2 El proceso de desarrollo 5

3.3 El producto del desarrollo 6

4 Evaluación 7

4.1 Proceso de evaluación 7

4.2 Análisis de resultados 7

Conclusión 8

Aportaciones realizadas 8

Trabajos futuros 8

Problemas encontrados 8

Opiniones personales 8

Lista de referencias 8

ANEXO A: Control de Versiones 9

ANEXO B: Seguimiento de proyecto fin de carrera 10

ANEXO C: Seguimiento de proyecto fin de carrera 11

ANEXO D: Manual de usuario 13

ANEXO E: Cuestionarios de evaluación 15

Índice de figuras

**No se encuentran elementos de tabla de ilustraciones.**  
En el documento, seleccione las palabras que desee incluir en la tabla de contenido y, en la pestaña Inicio, en Estilos, haga clic en un estilo de título. Repita el procedimiento para cada título que desee incluir y, a continuación, inserte la tabla de contenido en el documento. Para crear manualmente una tabla de contenido, en la pestaña Elementos de documento, en Tabla de contenido, seleccione un estilo y haga clic en el botón de flecha abajo. Haga clic en uno de los estilos en Tabla de contenido manual y, a continuación, escriba las entradas manualmente.

Índice de tablas

**No se encuentran elementos de tabla de ilustraciones.**  
En el documento, seleccione las palabras que desee incluir en la tabla de contenido y, en la pestaña Inicio, en Estilos, haga clic en un estilo de título. Repita el procedimiento para cada título que desee incluir y, a continuación, inserte la tabla de contenido en el documento. Para crear manualmente una tabla de contenido, en la pestaña Elementos de documento, en Tabla de contenido, seleccione un estilo y haga clic en el botón de flecha abajo. Haga clic en uno de los estilos en Tabla de contenido manual y, a continuación, escriba las entradas manualmente.

Glosario de términos

Catálogo de términos específicos del contexto del trabajo.

Introducción

Actualmente para la organización de aulas en la facultad de ingenierías, se utilizan alrededor de unos 500 calendarios de Google diferentes, que almacenan las horas de clase de cada una de asignaturas pertenecientes a los distintos grados y másteres que tiene ofertados la facultad. Estos calendarios, se ven reflejados en la página oficial por cursos, utilizando un calendario espejo, que tan solo tiene permisos de visualización sobre las asignaturas que pertenecen al curso que define dicho calendario.

Google Calendar, nos ofrece una herramienta potente y muy eficaz para mostrar eventos independientes, sin embargo, no es tan eficaz si lo que queremos es que no se permita la colisión de horarios según la ubicación de dichos eventos. Por otra parte, esta herramienta nos permite compartir horarios de manera muy sencilla con gran número de usuarios, por lo tanto, el objetivo de este trabajo es desarrollar una aplicación que permita la edición de todos estos horarios en una interfaz centrada en la distribución de espacios por hora y no tanto en los eventos individuales como tal, respetando en última instancia los calendarios actualmente existentes y volcando la información sobre ellos, pero teniendo siempre la posibilidad de realizar el proceso contrario, de importar información actualizada directamente sobre Google Calendar en caso de necesidad.

El objetivo de la aplicación por tanto, es asegurar que en ningún momento se produzca una colisión de dos horas de clase en un mismo espacio y permitiendo corregirla en caso de existir actualmente en los horarios. La visualización de estos se realizará diariamente, mostrando una lista de espacios, y las horas en las cuales están libres para su posible reserva, la cual únicamente se podrá realizar desde determinadas cuentas de usuario.

El acceso a la aplicación estará restringido a todas aquellas cuentas pertenecientes a la organización de la universidad de león realizando una identificación a través del proveedor de esta y solo ciertos usuarios tendrán acceso a la realización de reservas, mientras que la mayoría únicamente podrán visualizar los horarios en tiempo real de la escuela.

Objetivos

Por lo tanto, los objetivos de la aplicación a desarrollar están claramente definidos. En un primer lugar, debemos asegurar la consistencia del sistema verificando que en ningún momento se produzca una colisión durante la reserva de espacios y que, en caso de existir, pueda ser resuelta de forma rápida. Por otra parte, la aplicación debe poseer integración de los calendarios de Google, dejando reflejado en ellos toda la información que se encuentra en el sistema, y además, actualizándose a sí mismo desde estos en caso de que suceda algún cambio en los calendarios del propio Google.

En cuanto a la interfaz y el intercambio de información con el usuario, el sistema deberá de ser lo más transparente posible a este, con una interfaz intuitiva y fácil de usar, asegurando que la información de los cambios realizados se vuelca correctamente en los calendarios finales, y permitiendo por lo tanto, la edición de reservas por parte de los administradores.

Como último objetivo opcional, se plantea el desarrollo de un sistema de reservas provisionales, para facilitar el trámite de reserva de espacios que actualmente se realiza en papel.

Metodología

Debido al origen del proyecto, y teniendo en cuenta que este pertenece a la rama de la ingeniería de software, es casi imposible cerrar un diseño a la primera para pasarlo a una fase de desarrollo sin que sufra modificaciones por el camino, bien tanto por parte del cliente, como del propio equipo de desarrollo, por esta razón, la metodología a utilizar será una metodología ágil que acepte cambios en el software y en los objetivos iniciales del proyecto, adaptándose así a unas necesidades del cliente que pueden surgir durante la fase de desarrollo del proyecto y enfocándose en satisfacer las necesidades de este.

Se realizarán reuniones semanales sujetas a la disponibilidad del propio cliente y del equipo de desarrollo, cuyo objetivo es el de dar una visión del avance del proyecto al propio cliente, obteniendo “feedback” y ajustándolo a sus necesidades. Estas reuniones nos ayudarán a la creación de prototipos cambiantes, a los que se les irá añadiendo funcionalidad a lo largo del desarrollo, revisando y mejorando las partes del producto que necesiten algún cambio.

Esta metodología iterativa e incremental nos da una visión de avance del desarrollo, permite manejar posibles riesgos del proyecto y dando una mayor experiencia al equipo de desarrollo aumentando la productividad de este y generando un software de calidad.

# Estado del arte

# Planificación

Se realiza una planificación por objetivos, disponiendo en todo momento de un prototipo funcional, que irá creciendo a lo largo del desarrollo, cuyos cambios se verán reflejados en un repositorio de un controlador de versiones como es GitHub, pudiendo rectificar cualquier cambio en el proyecto gracias a este.

Estos objetivos se irán logrando semanalmente, permitiendo una ampliación de plazo en caso de ser necesario. Un listado de las distintas fases por las que ha pasado el desarrollo, cuyos objetivos están especificados en la parte derecha, sería el siguiente:

* 11 de febrero – 25 de febrero: Planteamiento inicial.
* 25 de febrero – 28 de febrero: Servidor Back-End.
* 28 de febrero – 10 de marzo: Autenticación OAuth2.
* 10 de marzo – 15 de marzo: Primer Front-End.
* 15 de marzo – 24 de marzo: Sistema de Reservas.
* 24 de marzo – 31 de marzo: Funcionalidades Front-End.
* 31 de marzo – 1 de mayo: Reforma del sistema de modelos.
* 1 de mayo – 19 de mayo: Reforma del sistema de IDs.
* 19 de mayo – 26 de mayo: Sistema de auto actualización.
* 26 de mayo – 5 de junio: Sistema de corrección de reservas.
* 5 de junio – 17 de junio: Creación configuración JSON.

Al final de cada una de estas fases, se obtiene un prototipo que con sus limitaciones, es completamente funcional para ser mostrado al cliente y estar sujeto a futuros cambios que este necesite, quedando reflejado en el repositorio online del propio proyecto en todo momento.

## Tareas a realizar

Para describir las tareas listadas anteriormente, se agruparan en tres grandes grupos, que serán resumidos a continuación conteniendo las ideas principales de las tareas que contienen, que serán descritas en partes posteriores del documento más detalladamente.

### Planteamiento Inicial

En esta primera fase, se realizan reuniones para sentar las bases de la aplicación y cuál será el principal problema a solucionar, discutiendo las distintas partes de esta, construyendo al desarrollador una idea de la posible solución que pudiera tener. En ella, el desarrollador decide que tecnologías utilizar durante las siguientes fases del desarrollo.

### Desarrollo de la interfaz

En esta fase intermedia, después de haber desarrollado un primer prototipo básico, y haber sentado un poco la base de lo que será en el futuro el servidor de la aplicación y la autenticación de usuarios que este utiliza, el desarrollador se reúne con el cliente para decidir junto a este cómo será la interfaz de usuario inicial, sujeta a cambios en el futuro, pero que será la base del prototipo final.

### Refactorización

Esta podría ser la fase final del desarrollo del primer prototipo completo y funcional de la aplicación, en ella se ha tenido que realizar una refactorización del código del servidor de back-end casi completa a causa de unos requisitos introducidos por el cliente, que conllevan la utilización de la API de Google Calendar. Si bien en un primer momento era conocida por el desarrollador la necesidad de llegar a utilizar esta API, no es hasta esta fase cuando realmente se crea esa necesidad por parte del cliente.

### Configuración

Una vez finalizada la refactorización, y el desarrollo de las últimas tareas en la aplicación como tal, se pasa a una última fase de configuración, en la cual se creará un archivo de configuración que contiene una lista de las carreras que serán mapeadas en la aplicación, así como las asignaturas que estas contienen y su id de Google Calendar, para futuros accesos de las cuentas robot que la aplicación contiene para la actualización de los calendarios en tiempo real. Además, se añade una lista de espacios que se verán reflejados en las tablas una vez iniciada la aplicación, así como una pequeña lista de configuraciones iniciales, que por ahora, solo contiene la fecha desde la cual se comenzaría a mapear el horario, y que podría ser actualizada anualmente por los administradores del servidor web.

## Diagrama de Gantt

Se adjunta a continuación un diagrama de Gantt que muestra en una línea de tiempo las diferentes tareas realizadas que previamente hemos descrito, indicando cuando se ha iniciado y finalizado cada una de ellas hasta la finalización del proyecto.

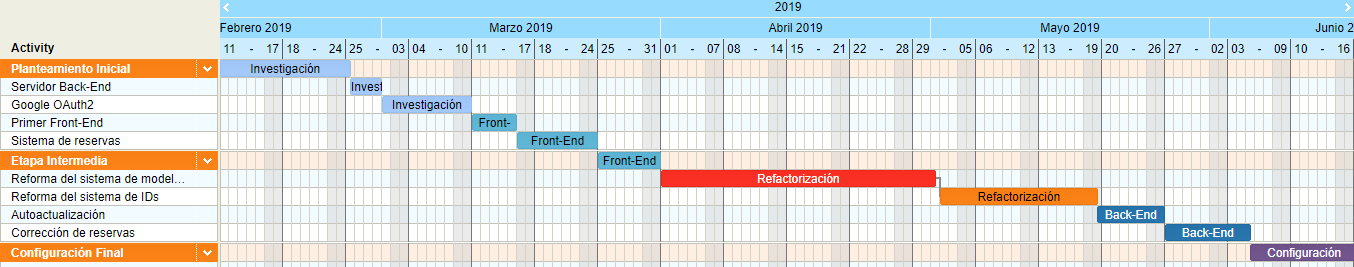


Figura 2.1 Diagrama de Gantt

Se podrá consultar una versión completa y de mayor resolución en el siguiente enlace: [Diagrama de Gantt](https://github.com/acampc01/TFG/blob/master/Gantt%20Completo.png). Las tareas mostradas en el diagrama, se verán reflejadas a los largo de las historias de usuario, en el capítulo sobre la metodología en este mismo documento.

# Tecnologías utilizadas

A continuación, se realizará una descripción de las tecnologías utilizadas durante el desarrollo de la aplicación comenzando por las diferentes tecnologías que conforman desde servidor de back-end hasta el front-end de la aplicación, y una pequeña parte final de configuración.

## Servidor Back-End

Como servidor web utilizaremos un Tomcat incluido en un proyecto Java Maven configurado gracias al Framework Spring, que junto a un conjunto de dependencias, proporcionan una herramienta completa para el desarrollo de nuestra aplicación web. Para la creación de nuestro proyecto Maven, utilizaremos “*Spring Initializr*”, disponible en este [enlace](https://start.spring.io/), que genera automáticamente el proyecto Maven con las dependencias especificadas, en nuestro caso, las listadas a continuación:

1. Spring Web Starter
2. Spring Security
3. Spring Data JPA
4. Spring DevTools
5. Thymeleaf
6. MySQL Driver
7. OAuth2 Client

Si bien vamos a explicar las más importantes en los apartados siguientes, hay que hacer mención a dependencias como “*Spring DevTools”*, que a pesar de no aportar nada al proyecto en sí, mejoran increíblemente la experiencia de desarrollo del mismo, permitiendo un reinicio y re compilación rápido de nuestro proyecto, durante la fase de codificación del mismo. Además, una última mención a “*MySQL Driver”*, que permite la interacción entre “*Spring Data JPA*” y la base de datos MariaDB utilizada en este proyecto.

### Spring Boot

Es el punto de partida para cualquier aplicación basada en Spring Framework, diseñado para acelerar el proceso de lanzamiento de aplicaciones web en fase de producción, permitiendo una construcción en segundos de un servidor web acompañado de distintas características añadidas por las dependencias adicionales. Es la base de nuestro proyecto, y la parte más importante, junto a las dependencias de seguridad de autenticación.

#### Spring Boot Security y OAuth2

Ambos forman nuestro framework de seguridad, potente y personalizable, que permite añadir un control de acceso y de autenticación a nuestro servidor web, si bien solo “*Spring Boot Security*” ya permite estas dos cosas, la dependencia de OAuth2, nos permite realizar la autenticación desde el servidor de una tercera entidad, en este caso, pasamos por Google OAuth2.0, que realizará una autenticación de los usuarios que accedan a nuestro servidor, asegurando que pertenecen a la organización de Google correspondiente con la Universidad de León. Volviendo a la dependencia de seguridad, esta nos permite realizar un control de acceso por rol de usuario, limitando las características del front-end según el tipo de usuario identificado en nuestro servidor.

Hasta ahora, tenemos un sistema que nos autentica la identidad del usuario que accede a la aplicación (Google OAuth2.0) y un sistema que controla qué podemos ver una vez que nuestro servidor ya tiene la confirmación de Google. En la parte perteneciente al Front-End, veremos una herramienta que en parte, acompaña a estas limitaciones de acceso por rol, llamada “*Thymeleaf*”.

#### Spring Data JPA

Antes de explicar de qué trata la dependencia de Data JPA, debemos explicar que nos aporta su antecesor Spring Data. Spring Data nos da un modelo de programación basado en Spring para el acceso a datos, permitiendo el intercambio de información con base de datos de cualquier tipo, así como la creación de aplicaciones para servicios en la nube, etc. Es un framework que contiene muchísimos subproyectos, según el tipo de intercambio que deseemos, en nuestro caso, un intercambio con una base de datos relacional de tipo MySQL.

Spring Data JPA es uno de los subproyectos más grandes pertenecientes a Spring Data, y nos facilita la implementación de repositorios JPA de una forma increíble, permitiendo la creación de peticiones a base de datos con tan solo la implementación de la cabecera de los métodos en una clase interfaz java, reduciendo el esfuerzo y el tiempo que esto conlleva. Para ilustrar esta explicación, se adjunta una captura de la clase repositorio perteneciente al modelo Usuario en nuestro proyecto:

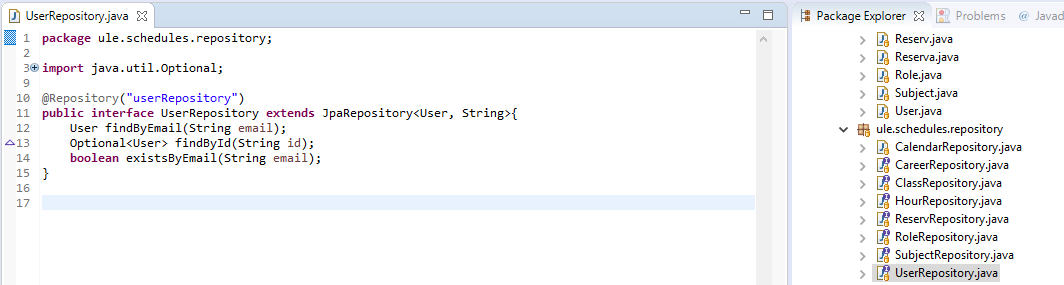


Figura 3.1 User Repository Interface

Como se puede observar, con tan solo una sintaxis específica en los nombres de los métodos, se puede obtener un intercambio satisfactorio con nuestra base de datos, gracias a que es Spring Data JPA el encargado de la creación de las peticiones.

### Google Calendar API

Otro de los grandes pilares sobre los que trabaja nuestra aplicación, permitiéndonos intercambiar información con los calendarios de google específicos de cada asignatura de la facultad, esto logra la API de Google Calendar. Si bien gracias a Spring Boot y sus dependencias, podríamos crear una aplicación web completamente independiente, en nuestro caso y a causa del diseño elegido, utilizaremos esta API para aportar información a nuestra aplicación.

La API de Google Calendar, permite obtener tanto los calendarios como los eventos de estos a los que se tiene acceso desde una cuenta de google, bien sea una personal, o una cuenta robot, como vendría a ser en nuestro caso, integrando Google Calendar en nuestra aplicación.

Como en la facultad de ingenierías, a la que está enfocado este proyecto, posee alrededor de 550 asignaturas con su correspondiente calendario, y el conjunto de estos calendarios tiene alrededor de 13000 eventos, y debido a las limitaciones de la API, que nos limita a 250 calendarios por cuenta, y a 2500 eventos por calendario, nos veremos obligados a crear dos cuentas robot, una para las asignaturas correspondientes a los diferentes grados, y otra para las asignaturas correspondientes a los másteres que desea mapear la facultad, a las cuáles se les otorgará permiso de edición en los calendarios.

Una vez otorgados estos permisos, la API nos permitirá obtener un listado de calendarios y un listado de eventos por calendario, recolectando la información necesaria para la creación de las reservas en nuestra aplicación. Este recolectado, se realiza distinguiendo dos campos dentro de un evento cualquiera de la API, la fecha de inicio y finalización del evento, y su campo de localización. Por especificaciones del cliente, los eventos deben comenzar y finalizar en horas a en punto o y media, así como tener en el apartado de ubicación una cadena de caracteres perteneciente a un listado predefinido de aulas con nombre y numeración o simplemente tener este apartado vacío, en cualquier otro caso, el evento cuya ubicación sea desconocida, será ignorado por el sistema de reservas. Todo esto, da lugar a un filtrado de eventos, que nos deja con alrededor de unos 9800 eventos con los que trabajar dentro del sistema de reservas, todos ellos numerados por la API de calendar, siendo editables o eliminables individualmente.

## Cliente Front-End

### Thymeleaf

### Bootstrap

### Material Design

# Evaluación

Demostración de la validez de la solución elaborada. La solución se considera válida si resuelve los problemas expuestos en el planteamiento del problema y satisface los objetivos definidos en la introducción. Según el caso la forma de evaluación se basará en la ejecución de casos de prueba o en la realización de cuestionarios. Extensión entre quince y veinte páginas.

## Proceso de evaluación

### Forma de evaluación

Explicar la forma en la cual se ha evaluado la aplicación

### Casos de prueba

Casos de pruebas realizados

## Análisis de resultados

Conclusión

Expresión personal del conjunto de conclusiones que, a juicio del autor, se derivan de los resultados expuestos en el trabajo. Deberá tener una extensión entre cinco y diez páginas.

Aportaciones realizadas

Trabajos futuros

Problemas encontrados

## Opiniones personales

# Lista de referencias

Información bibliográfica citada en el texto del trabajo. Otras lecturas recomendadas o consultadas, de figurar, aparecerán en anexos.

Se debe seguir la norma ISO 690 (buscar en google ISO 690 ugr)

ANEXO A: Control de Versiones

ANEXO B: Seguimiento de proyecto fin de carrera

ANEXO C: Seguimiento de proyecto fin de carrera

**Obligatorio.** Seguimiento del trabajo real.

### Forma de seguimiento

### Planificación inicial

### Planificación final

ANEXO D: Manual de usuario

Si el trabajo ha consistido en la elaboración de una aplicación se incluirá el manual de usuario de la misma.

ANEXO E: Cuestionarios de evaluación

Cuestionarios utilizados durante la fase de evaluación.