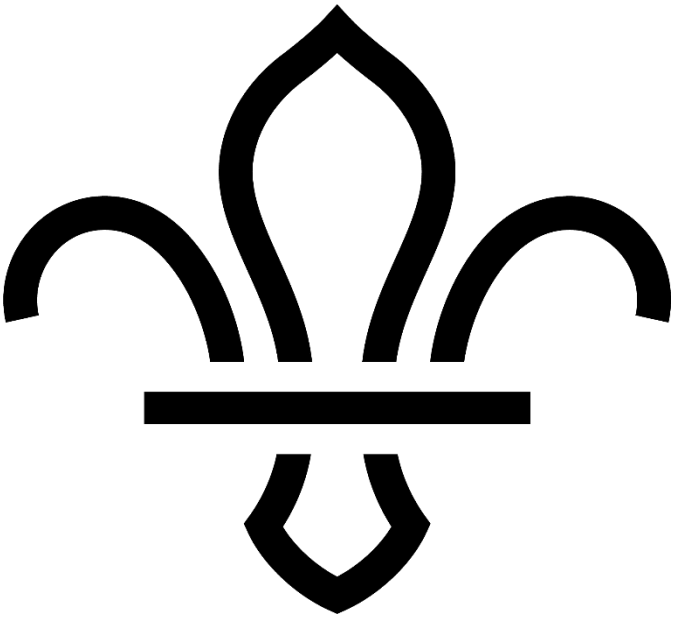
Gestor de Grupos Scout



Bootstrap

Por:
Miguel Negrón
Adrián Rubio

Índice

Descripción del proyecto

El proyecto se basa en crear una aplicación que facilite la gestión de los grupos Scout utilizando el framework de Java Spring.

Justificación del proyecto

La idea de desarrollar esta aplicación surge a raíz de que actualmente estos no disponen de ninguna herramienta diseñada específicamente para ellos, lo cual hace que suelan gestionarse mediante combinaciones de servicios muy poco eficientes como Google drive, Dropbox, Mega, hojas de cálculo...

Esto nos brinda la oportunidad de coger un sistema de datos real y crear una aplicación que los gestione de manera más eficiente.

Recursos

Spring (Dependencias)

Spring es un framework para el desarrollo de aplicaciones y contenedor de inversión de control, de código abierto para la plataforma Java.

El corazón de Spring Framework es su contenedor de inversión de control. Su trabajo es instanciar, inicializar y conectar objetos de la aplicación, además de proveer una serie de características adicionales disponibles en Spring a través del tiempo de vida de los objetos.

Los objetos creados y gestionados por el contenedor se denominan objetos gestionados o Beans. Estos objetos son del tipo POJO (Plain Old Java Object). Para realizar su tarea el contenedor necesita información indicando cómo instanciar y conectar entre sí los beans.

- Spring Boot
- Spring starters generator
- Spring Security
- Spring Session
- Spring Web
- Spring JPA
- Hibernate
- WebJars
- Servidor Tomcat

Base de datos H2

Al estar desarrollando la aplicación hemos decidido utilizar H2 por la gran comodidad que supone el tener la base de datos en ficheros. Obviamente a la hora de desplegar esta aplicación se cambiaría la base de datos a una más adecuada, como por ejemplo MySQL.

Al estar utilizando Maven y Spring (Sin consultas nativas de H2) este cambio solo consistiría en sustituir el artifact de H2 por el de MySQL en la configuración de Maven (pom.xml) junto a unos pequeños ajustes en las credenciales y la ruta de la base de datos en los ajustes de Spring.

Frontend

- Thymeleaf: es una biblioteca Java que implementa un motor de plantillas HTML que puede ser utilizado tanto en modo web como en otros entornos.

  Se acopla muy bien para trabajar en la capa vista del MVC, o modelo vista controlador, de aplicaciones web, pero puede procesar cualquier archivo XML, incluso en entornos desconectados.

- Bootstrap: es una biblioteca multiplataforma o conjunto de herramientas de código abierto para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basados en HTML Y CSS, así como extensiones de JavaScript adicionales.

  A diferencia de muchos frameworks web, solo se ocupa del desarrollo frontend.

- HTML
- CSS
- JQuery: es una biblioteca multiplataforma de JavaScript, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

  Pese a no utilizarlo directamente, ha de estar presente para que Bootstrap pueda funcionar.

Desarrollo

Aprendizaje de las tecnologías

Durante todo el proyecto hemos tenido que hacer numerosas pausas para aprender con más profundidad diferentes aspectos de las tecnologías que estábamos usando en nuestro proyecto. Aunque han estado presentes desde el comienzo hasta el final, por simplicidad las nombraremos todas aquí.

Investigación inicial

Al empezar el proyecto decidimos utilizar el Framework Spring. El hecho de que ninguno de los integrantes conocíamos este framework hacía imprescindible el comenzar con un exhaustivo aprendizaje de los sus aspectos más importantes:

- Arquitectura: Spring basa su metodología de trabajo en el modelo-vista-controlador (MVC) por lo que debemos fragmentar nuestra programación en clases específicas para diferentes tareas. Estas clases se irán describiendo con más profundidad a lo largo del proyecto pero fundamentalmente se podrían agrupar en:
    - Modelo: Clases utilizadas para generar la base de datos mediante Hibernate y los diversos servicios para interactuar con esta.
    - Vista: Diferentes plantillas de Thymeleaf (HTML modificado), CSS y JavaScript usadas para servir la interfaz.
    - Controlador: Clases encargadas de despachar peticiones al servidor y servir las vistas correspondientes.

- Estructura: Desde el comienzo de la investigación descubrimos que si bien Spring no requiere ninguna estructura de proyecto específica, sí que hay un claro estándar que facilita enormemente el desarrollo de un proyecto Spring. No solo porque el funcionamiento predeterminado busca ciertos elementos en ciertos sitios, lo cual se puede cambiar fácilmente si así lo requiriésemos. Otras ventajas menos llamativas pero igual de presentes han sido, por ejemplo, que aunque fuéramos dos personas con métodos de trabajo distintos y sin posibilidad de tener reuniones físicas, al tener que ceñirnos a una estructura ya predefinida, siempre sabíamos con certeza el lugar donde debían ir nuevos elementos o cambios que necesitáramos hacer. O la búsqueda de errores y código de ejemplo en internet, ya que al seguir el estándar se elimina una capa de complejidad al tener que integrar nuestros hallazgos en el proyecto.

Profundizar en el uso de plantillas

A la hora de implementar la interfaz gráfica del proyecto nos dimos cuenta de que lo que habíamos aprendido sobre Thymeleaf, la parte encargada del frontend de Spring, era bastante elemental. Esto nos llevó a buscar más información sobre la programación dentro de Thymeleaf y cómo incorporarla a la estructura de nuestros datos.

También tuvimos que repasar un poco Bootstrap, e investigar cuál era la manera óptima integrarlo en nuestro servidor, la cual llegamos a la conclusión de que era unas librerías llamadas Webjars las cuales permiten integrar todas las librerías de Bootstrap y JQuery (entre muchas otras) en el periodo de compilación.

Repaso para diferentes situaciones con el ORM

Lidiando con las diferentes relaciones Uno a Muchos, y Muchos a Muchos que hay en nuestro proyecto tuvimos que ver varias explicaciones de cómo integrar ciertos tipos de estas relaciones en nuestro proyecto. Esto nos ayudó a aplicarlas después en las plantillas ya que gracias a ciertas prácticas que descubrimos en un tutorial de hibernate permitían obviar mucho código en Thymeleaf.

Extracción y normalización de la estructura de datos de Dropbox

Para la generación de nuestra estructura de datos partimos de un sistema ya existente, dado que Miguel pertenece a un grupo Scout en el que se organizaban mediante varios archivos Excel.

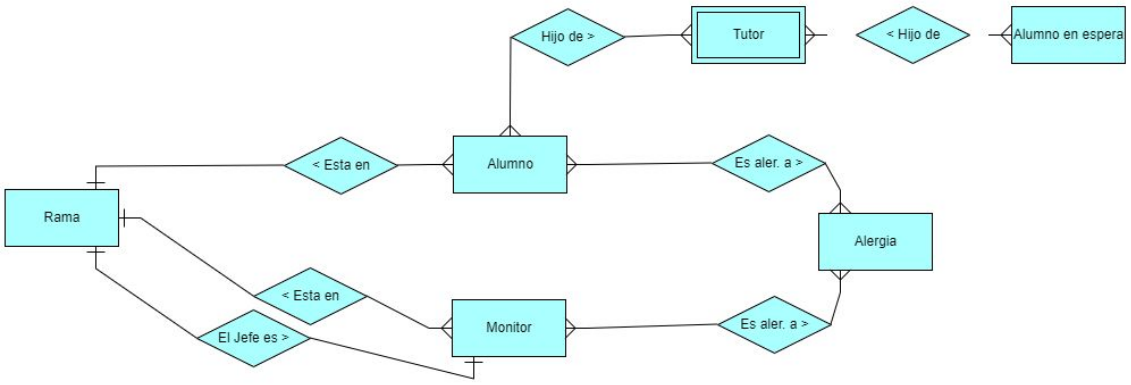
Esto era una manera bastante obsoleta de organizar los datos, sobre todo teniendo en cuenta que no seguían un orden demasiado útil a la hora de extraer datos concretos.

Por eso pensamos que sería una buena idea crear una aplicación para ordenar todo de una manera más sencilla y ordenada. (Las funcionalidades vendrían más adelante)

Lo primero fue extraer los integrantes de este grupo Scout, es decir, partir de que había niños, tutores, monitores, etcétera.

Una vez hecho esto desarrollamos un modelo entidad relación sencillo, para poder ver las cosas más claras y separar las entidades mientras les dábamos sus atributos.

Después de acabar con las entidades y las relaciones, solo quedaba normalizar los datos para eliminar redundancias.



Una vez normalizados los datos, ya podríamos empezar a trabajar y generar nuestras clases a partir del esquema.

Estructuración del código:

Paquete principal (com.pio2.spring)

En este paquete existe la clase Pio2Application, que básicamente es la clase principal desde donde se inicializa el proyecto.

Esta es la clase que usamos en un principio para poder insertar entidades automáticamente al iniciar la aplicación y comprobar que funcionaba correctamente.

Paquete de configuraciones (com.pio2.spring.configuracion)

En este paquete residen clases marcadas con la anotación @Configuration las cuales sirven para configurar aspectos no globales de la aplicación, ya que estos se pueden configurar en el archivo application.properties.

Paquete de controladores (com.pio2.controladores)

Este paquete contiene controladores para las distintas entidades que se manejan, estos son clases que se encargan mediante los métodos @GetMapping y @PostMapping de generar los modelos y de redirigir a un HTML u otro según donde estemos clicando y las funcionalidades que necesitemos.

También son importantes estas clases para poder construir los formularios y guardar los datos de las entidades que queramos añadir desde la aplicación.

Paquete de entidades (com.pio2.spring.entidades)

Este es el paquete que contiene todas las clases que hemos construido , es decir los objetos o entidades de los que queremos almacenar datos y poderlos modificar para hacer una buena gestión de los mismos.

Por el momento tenemos las clases Alergia, Curso, Monitor, Niño y Tutor. Esto es ampliable de tal manera que en un futuro podrían crearse más clases como por ejemplo una clase para gestionar las listas de espera para ingresar en el campamento.

Paquete de "Enums" (com.pio2.spring.enums)

En enum o enumeración es una lista de constantes con nombre que definen un nuevo tipo de datos. Un objeto de un tipo de enumeración sólo puede contener los valores definidos por la lista. Por lo tanto, una enumeración le brinda una manera de definir con precisión un nuevo tipo de datos que tiene un número fijo de valores válidos.

Este tipo de variable que podemos utilizar en Java, nos ha servido especialmente para definir las posibles alergias de los niños, los cursos que pueden existir dentro del campamento y los cargos que pueden tener los monitores.

Paquete de repositorios (com.pio2.spring.repositorios)

Estas clases se encargan de crear un repositorios para cada clase, los cuales heredan los métodos CRUD entre otros de una interfaz : JPA.repository.

Esto habilita la creación de consultas más complejas de manera más sencilla a nivel de código, como puede ser: public Curso findByAtributoOrderXXX(String atributo). Es decir, sin código interno, Spring procesa la petición solo con la sintaxis del nombre del método

Paquete de seguridad (com.pio2.spring.seguridad)

Aquí disponemos de la configuración de seguridad de nuestra aplicación. Por el momento tiene unas credenciales sencillas para facilitar la entrada al índice,(admin, admin como usuario y contraseña) Lo que podría ser más adelante un login normal en el que se use la base de datos para registrar los e-mails y las contraseñas.

En este paquete, también necesitaremos especificar a qué formulario queremos vincular el login y a dónde queremos ir después, así como quién está autorizado a entrar y quién no y quien puede hacer que cosas y que cosas no.

Paquete de servicios (com.pio2.spring.servicios)

En este paquete se encuentran los servicios de cada clase principal, y los utilizamos para describir los métodos de creación, edición y búsqueda que vamos a emplear para cada cosa con respecto a la base de datos.

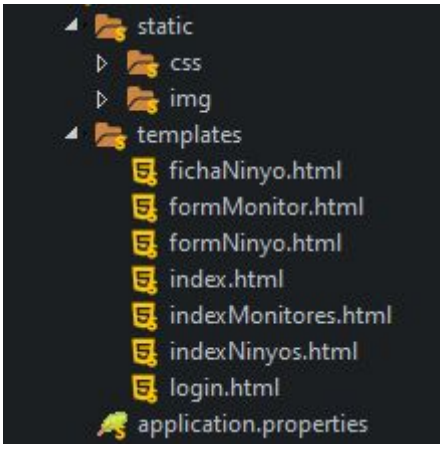
Paquete de utilidades (com.pio2.spring.utilidades)

Aquí encontraremos diferentes funcionalidades menos específicas, como puede ser una validación del DNI, para comprobar que existe mediante la fórmula correspondiente.

O la clase Inicializador de datos, que inicializa una lista de datos para que nuestra base de datos no esté vacía cuando lanzamos la aplicación y tengamos listados en los que mirar que todo funciona correctamente.

Carpeta de recursos HTML y configuración

En esta carpeta tenemos los HTML necesarios para toda la aplicación web, es decir, consta del Login, un índice, y los diferentes formularios para la introducción y modificación de datos.



También existe un archivo, application.properties ,de configuración general con algunas propiedades globales que necesitábamos implementar. Algunas de ellas son por pura utilidad a la hora de desarrollar, como el mostrar el código sql por consola o que thymeleaf no guarde en cache las plantillas (para poder recargar la pagina y ver cambios en el html recién guardados sin necesidad de reiniciar el servidor).

```
1
2  # Puerto de la aplicación
3  server.port=9000
4  # Configuración del almacenamiento de sesiones con Redis
5  #spring.session.store-type=redis
6
7  # URL jdbc de conexión a la base de datos
8  spring.datasource.url=jdbc:h2:./pio2
9
10 # Usuario y contraseña de la base de datos
11 spring.datasource.username=sa
12 spring.datasource.password=
13
14 # Habilitamos la consola de H2
15 # http://localhost:{server.port}/h2-console
16 # En nuestro caso http://localhost:9000/h2-console
17 spring.h2.console.enabled=true
18
19 # Habilitamos los mensajes sql en el log
20 spring.jpa.show-sql=true
21
22
23 spring.thymeleaf.cache=false
24
25
26
```

Bibliografía

- López L.M. (Sin fecha)

  Curso de Spring Core 5

  https://openwebinars.net/academia/aprende/spring-core/

- López L.M. (Sin fecha)

  Curso de Spring Boot y Spring MVC 5: Creando una aplicación con Spring Boot y Spring MVC

  https://openwebinars.net/academia/aprende/spring-boot/

- López L.M. (Sin fecha)

  Curso de Introducción a Thymeleaf

  https://openwebinars.net/academia/aprende/introduccion-thymeleaf/

- López L.M. (Sin fecha)

  Curso de Thymeleaf intermedio

  https://openwebinars.net/academia/aprende/thymeleaf-intermedio/

- López L.M. (Sin fecha)

  Curso Online de Hibernate y JPA

  https://openwebinars.net/academia/aprende/hibernate/

- Pérez J.D. (Sin fecha)

  Bootstrap 4: Maquetación Responsive y Layout

  https://openwebinars.net/academia/aprende/bootstrap-4-layout/

- Rod Johnson, Juergen Hoeller, Keith Donald, Colin Sampaleanu, Rob Harrop, Thomas Risberg, Alef Arendsen, Darren Davison, Dmitriy Kopylenko, Mark Pollack, Thierry Templier, Erwin Vervaet, Portia Tung, Ben Hale, Adrian Colyer, John Lewis, Costin Leau, Mark Fisher, Sam Brannen, Ramnivas Laddad, Arjen Poutsma, Chris Beams, Tareq Abedrabbo, Andy Clement, Dave Syer, Oliver Gierke, Rossen Stoyanchev, Phillip Webb, Rob Winch, Brian Clozel, Stephane Nicoll, Sebastien Deleuze, Jay Bryant, Mark Paluch (2020)
  Spring Framework Documentation
  https://docs.spring.io/spring/docs/current/spring-framework-reference/

- Equipo de desarrolladores de Bootstrap - https://github.com/orgs/twbs/people (2020)
  Documentación de Bootstrap 4

https://getbootstrap.com/docs/4.1/getting-started/introduction/

- Equipo de w3schools (2020)
  Tutorial de HTML
  https://www.w3schools.com/html/default.asp

- Equipo de w3schools (2020)
  Tutorial de CSS
  https://www.w3schools.com/css/default.asp

- Oracle (Sin fecha)
  Documentación de Java 8
  https://docs.oracle.com/javase/8/docs/api/

- Usuarios de StackOverflow
  Diferentes consultas de errores
  1. https://stackoverflow.com/questions/13027214/what-is-the-meaning-of-the-cascadetype-all-for-a-manytoone-jpa-association
  2. https://stackoverflow.com/questions/16246675/hibernate-error-a-different-object-with-the-same-identifier-value-was-already-a/16246858
  3. https://stackoverflow.com/questions/46057873/how-to-avoid-javax-persistence-entityexistsexception-different-object-with-the