Práctica 1
Programación Web

Mohssin Bassat Sidki
Javier Castilla Arroyo
María José García Aragón
Alejandro Gómez Amaro

Índice

Repositorio en GitHub → https://github.com/alegomezamaro/pw2526-gm3-1

1. Introducción

El proyecto desarrollado consiste en una aplicación web basada en Spring Boot, empleando Thymeleaf para la capa de presentación y MySQL como sistema gestor de base de datos.

El objetivo principal ha sido implementar un sistema que permita gestionar los distintos elementos del club náutico: socios, inscripciones, embarcaciones, familias y reservas, ofreciendo funcionalidades de alta, consulta, actualización y filtrado.

La aplicación se ha estructurado siguiendo un enfoque MVC (Modelo-Vista-Controlador), lo cual ha permitido una mejor separación de responsabilidades y una mayor claridad durante el desarrollo.

A lo largo del proceso, el equipo ha tenido que integrar diferentes capas tecnológicas, configurar la conexión con MySQL, diseñar repositorios compatibles con las propiedades SQL proporcionadas por la práctica y ajustar la comunicación entre controladores y vistas.

La experiencia ha sido especialmente relevante para comprender cómo funciona un proyecto completo en Spring Boot, abordando desde la lógica de negocio hasta la interacción con la base de datos.

2. Diseño y estructura del proyecto

La estructura del proyecto se basó en tres paquetes principales:

1. **model** → Contiene las clases de dominio (Socio, Embarcación, Inscripción, etc.) y sus respectivos repositorios.

2. **controller** → Incluye los controladores Spring MVC responsables de gestionar las rutas, recibir parámetros y preparar los datos para las vistas.

3. **resources/templates** → Contiene todas las vistas Thymeleaf utilizadas por el usuario final.

2.1. Modelo de dominio

Cada entidad del proyecto representa un elemento gestionado por el club náutico:

- **Socio:** incluye atributos como DNI, nombre, apellidos, dirección, fechas y un booleano que indica si es patrón.
- **Embarcación:** recoge matrícula, nombre, tipo, plazas, dimensiones y patrón asignado.
- **Inscripción:** mantiene información sobre la cuota (individual o familiar), el titular, la familia asociada y la fecha.
- **Familia, Reserva, etc.:** estructuran las demás relaciones necesarias.

Decidimos estructurar nuestra práctica de esta forma ya que estas son las entidades mínimas necesarias para representar las funcionalidades que nos pedían; alta de socios, alta de inscripciones, alquiler de embarcaciones, gestión de familias, asignación de patrón y búsquedas y listados.

Se hizo especial hincapié en mantener consistencia entre todos los modelos, ya que una de las principales dificultades fue la coherencia de tipos entre clases y base de datos.

3. Dificultades encontradas

3.1. Inconsistencias entre clases Java y base de datos

Esta fue la primera gran dificultad. Se encontraron errores como:

- atributos declarados como String en Java, pero INT en MySQL,
- enumeraciones que no coincidían con los valores almacenados,
- fechas representadas de forma diferente (LocalDate, DATE, VARCHAR).

Estos fallos acabaron provocando inserciones que no se realizaban, errores en la página y búsquedas que no devuelven resultados. Se solucionó estandarizando los tipos y revisando todas las columnas de la base de datos.

3.2. Problemas con la conexión MySQL y el conector JDBC

El problema se debe a que el **código intenta insertar un campo que no existe en la base de datos MySQL**. El **conector JDBC** traduce ese insert a SQL nativo, pero al no encontrar dicha columna en el esquema actual, lanza la excepción que te avisada de la incongruencia.

3.3. Carga del fichero sql.properties

La carga del fichero SQL también causó problemas; teniamos rutas relativas mal establecidas, repositorios que se inicializaban antes de cargar propiedades y sobretodo, repositorios con sqlQueries == null.
Para corregirlo añadimos "(if (sqlQueries == null) createProperties())".

3.4. Comunicación entre controlador y vista

También afectaron errores como:nombres de atributos inconsistentes (newAlquiler, newInscripcion, etc.), vistas mal nombradas,con algún espacio o caracter oculto, poner la ruta del controlador ".html", vistas que esperaban un objeto que el controlador no enviaba.

4. Definición de clases

El diagrama representa el modelo de clases de un sistema para la gestión integral de un **Club Nautiuco**.

La clase **Socio** es la entidad central, almacenando datos personales y permitiendo operaciones como añadir socios o asignarles el título de patrón. La clase **Familia** vincula a un socio titular con otros miembros mediante el número de adultos y niños.

La clase **Inscripción** registra las cuotas anuales, su tipo y la relación con socios o familias. El módulo de navegación se estructura en las clases **Patrón** y **Embarcación**, que permiten gestionar los trabajadores y asignarlos a embarcaciones. El sistema también incorpora las clases **Alquiler** y **Reserva**, que gestionan la disponibilidad de embarcaciones, las fechas de uso y el coste asociado.

En conjunto, el diagrama refleja un sistema modular y bien organizado, donde cada clase cumple una función específica dentro de la operativa del club.