# Programación web 2025/2026 – Práctica 2

## Introducción y normativa general

La práctica 2 consiste en el diseño e implementación de una API REST para el acceso programático a algunos de los recursos de la aplicación web desarrollada en la práctica 1. Para ello se explicarán progresivamente los conceptos relacionados con la implementación de los métodos de la API que den respuesta a las distintas peticiones HTTP (GET, POST, PUT, PATCH y DELETE). En esta práctica continuará el aprendizaje de Java y Spring. También se debe continuar el control de versiones con Git, utilizando GitHub como plataforma de alojamiento y compartición del proyecto de prácticas.

Consideraciones importantes:

- El proyecto se realizará en equipos de cuatro personas, que se mantendrá invariable durante el curso académico.
- Se realizará un seguimiento semanal a través del repositorio en GitHub, valorando el trabajo continuo y la contribución individual de cada miembro del equipo.
- La práctica 2 supone el 40% de la calificación correspondiente a las entregas de prácticas (que representan el 30% de la asignatura). Se debe obtener una puntuación mínima de 3 puntos sobre 10 para poder aprobar las prácticas.
- Se entregará un informe en formato PDF (máximo 4 páginas) que deberá explicar y fundamentar las decisiones de diseño e implementación que ha adoptado el equipo, así como la relación de fuentes consultadas y un análisis de las dificultades encontradas.
- El código debe seguir las buenas prácticas de nombrado (variables, paquetes, etc.) y estar documentado haciendo uso de Javadoc a nivel de clase y de métodos: https://www.oracle.com/technical-resources/articles/java/javadoc-tool.html
- La propuesta de ejercicios tiene margen de decisión por parte del equipo. Dado que se espera que estas decisiones sean distintas por cada equipo de prácticas, deberán estar justificadas pertinentemente en el informe.
- Solo hará entrega de la práctica un miembro del equipo a través de Moodle, aquel que hizo la entrega de la práctica 1. Se deberá subir un archivo ZIP a la tarea Moodle con los proyectos VSC (API y código cliente) y el informe en PDF. El archivo ZIP que se entrega debe ser nombrado de la siguiente forma:

  GM<gm>_<login>.zip, donde <gm> es el número de grupo mediano y <login> es el nombre de usuario UCO de la persona del equipo que hace la entrega.

El incumplimiento de las normas de entrega implicará la consideración de práctica no entregada. El retraso en la entrega supondrá la no calificación en la práctica. La fecha de entrega se indicará en la tarea Moodle correspondiente.

**IMPORTANTE**: No se admitirá entrega alguna fuera de plazo o a través de un medio distinto a la tarea de Moodle (p.ej. envío por email).

## Aspectos generales

Los controladores para la API pueden implementarse en el proyecto de la práctica 1, pero deben estar perfectamente diferenciados en un paquete propio (por ejemplo, "*contoller.api*"). Si se prefiere crear un nuevo proyecto para el desarrollo de la API, se deben incluir los paquetes con las clases de dominio y los repositorios de la práctica 1.

Es posible que se deba adaptar o ampliar el diseño de la base de datos para la implementación de las nuevas funcionalidades. Se recomienda que no se modifique la estructura general de las tablas, tan solo se añadan nuevas columnas cuando sea necesario. De esta forma, se garantiza que el código de la práctica 1 siga siendo válido y pueda reutilizarse. Si se tienen que hacer cambios más profundos a la base de datos, se recomienda construirla en la cuenta de otro miembro del equipo para no dificultar el uso de la base de datos original durante la evaluación de la práctica 1.

En la medida de lo posible, deben reutilizarse las clases de la práctica 1, especialmente los métodos CRUD de los repositorios. Además, deberán aplicarse las mismas restricciones a las operaciones, salvo que expresamente se indique lo contrario.

## Semana 1: Implementación de API REST (GET y POST)

Aplicando los conceptos explicados en clase y siguiendo con las directrices de diseño, se debe implementar una API REST que permita acceder a los siguientes recursos:

**A. Gestión de socios**

La API debe proporcionar métodos para realizar las siguientes operaciones sobre socios:

1. Obtener la lista completa de socios, sean titulares o no, del club náutico (GET)
2. Obtener la información de un socio, sea titular o no, dado su DNI (GET)
3. Crear un nuevo socio sin asociación a ninguna inscripción previa (POST)
4. Crear un nuevo socio asociándolo a una inscripción familiar ya existente (POST)

Respecto a las inscripciones, se deben soportar las siguientes operaciones:

1. Obtener la lista de inscripciones individuales (GET)
2. Obtener la lista de inscripciones familiares (GET)
3. Obtener la información de una inscripción dado el DNI del socio titular (GET)
4. Crear una inscripción para un socio titular (POST)

**B. Gestión de flota y patrones**

La API debe proporcionar métodos para realizar las siguientes operaciones sobre flotas y patrones:

1. Obtener la lista completa de embarcaciones (GET)
2. Obtener la lista de embarcaciones según el tipo de embarcación (GET)
3. Obtener la lista completa de patrones (GET)
4. Crear una nueva embarcación, sin asociarle patrón (POST)
5. Crear un nuevo patrón, sin asociarle embarcación (POST)

**C. Alquiler de embarcaciones**

La API debe proporcionar los siguientes métodos para realizar operaciones sobre alquileres:

1. Obtener la lista completa de alquileres (GET)
2. Obtener la lista de alquileres futuros dada una fecha (GET)
3. Obtener la información concreta de un alquiler dado su identificador (GET)
4. Obtener las embarcaciones disponibles dada una fecha de inicio y de fin (GET)
5. Crear un alquiler para una embarcación disponible, sin incluir la vinculación de los socios que participan en ella (POST)

**D. Organización de actividades**

Respecto a la organización de actividades, la API debe proporcionar métodos para realizar las siguientes acciones:

1. Obtener la lista completa de reservas (GET)
2. Obtener la lista de reservas futuras dada una fecha (GET)
3. Obtener la información concreta de una reserva dado su identificador (GET)
4. Crear una reserva para una embarcación disponible (POST)

Finalmente, implemente uno o varios programas Java que realicen invocaciones a la API para demostrar su funcionamiento. Se deben considerar tanto peticiones que culminen en acciones exitosas como casos de error (datos inválidos, incumplimiento de restricciones, etc.)

## Semana 2: Implementación de API REST (PUT, PATCH y DELETE)

Se debe completar la implementación de la API REST para soportar operaciones de modificación y eliminación de recursos.

**A. Gestión de socios**

La API debe incluir métodos para realizar las siguientes operaciones sobre socios e inscripciones:

1. Actualizar los campos de información de un socio, excepto el DNI (PATCH)
2. Actualizar una inscripción individual para convertirla en una familiar (PUT)
3. Vincular a un nuevo miembro en una inscripción familiar, asumiendo que el nuevo miembro ya se ha registrado previamente como socio (PATCH)
4. Desvincular a un miembro de una inscripción familiar, sin borrar al socio (PATCH)
5. Cancelar una inscripción individual o familiar, dado el DNI del socio titular, sin borrar a los socios (DELETE)
6. Eliminar a un socio si no está vinculado a ninguna inscripción (DELETE)

**B. Gestión de flota y patrones**

La API debe incorporar métodos para realizar las siguientes operaciones sobre flotas y patrones:

1. Actualizar los campos de información de una embarcación, excepto la matrícula (PATCH)
2. Actualizar los campos de información de un patrón, excepto el DNI (PATCH)
3. Vincular un patrón a una embarcación (PATCH)
4. Desvincular un patrón de una embarcación (PATCH)
5. Eliminar una embarcación si no está vinculada a ningún alquiler o reserva, ya sea pasada o futura (DELETE)
6. Eliminar a un patrón si no está vinculado con ninguna embarcación (DELETE)

**C. Alquiler de embarcaciones**

La API debe proporcionar los siguientes métodos para realizar operaciones sobre alquileres:

1. Vincular a un nuevo socio (no titular) a un alquiler futuro, actualizando el coste y número de plazas (PATCH)
2. Desvincular a un socio (no titular) de un alquiler futuro, actualizando el coste y el número de plazas (PATCH)
3. Cancelar un alquiler que aún no se haya realizado (DELETE)

**D. Organización de actividades**

Respecto a la organización de actividades, la API debe incorporar nuevos métodos para realizar las siguientes acciones:

1. Modificar la fecha de una reserva futura a otra posterior, solo si la embarcación ya asignada está disponible en la nueva fecha solicitada (PATCH)
2. Modificar los datos de la reserva (descripción y número de plazas). En el caso del número de plazas, debe comprobarse que no exceda a la capacidad máxima de la embarcación asignada (PATCH)
3. Cancelar una reserva de actividad que aún no se haya realizado (DELETE)

Finalmente, implemente uno o varios programas Java que realicen invocaciones a la API para demostrar su funcionamiento. Se deben considerar tanto peticiones que culminen en acciones exitosas como casos de error (datos inválidos, incumplimiento de restricciones, etc.)