

# Guía del Proyecto en Equipo

## Desarrollo de una Aplicación Telemática

### Introducción

A lo largo de todo el curso se irá realizando un PROYECTO DE PROGRAMACIÓN DE APLICACIONES TELEMÁTICAS elegido por cada equipo de alumnos lo que va a permitir comprender y asimilar los conceptos estudiados.

Esta actividad es de suma importancia para la obtención de los objetivos del curso, ya que con ella se quiere conseguir que:

- El alumno sienta la necesidad de utilizar de manera sistemática algún proceso de desarrollo predefinido al enfrentarse a un problema de tal envergadura que no es posible abordarlo de otra manera.
- Al tener que aplicar todo lo que se les va explicando en teoría, vayan estudiando la teoría e interiorizándola.
- Adquieran experiencia en un trabajo en equipo (que requiere una gran colaboración grupal), analizando explícitamente a final de curso las ventajas que este método de trabajo tiene y sus desventajas, que serán los problemas que necesariamente hayan ido surgiendo en el desarrollo de la actividad.
- Sean capaces de abordar su proyecto de fin de carrera con mayor confianza dado que ya se han enfrentado a un proyecto de cierta envergadura.
- Experimenten directamente su futura realidad profesional.

Características globales:

- Equipos de 4 personas: Los equipos para la realización del proyecto se organizan según deciden los alumnos.
- Elección del Proyecto por parte del equipo. Todos los grupos realizarán las prácticas sobre la misma temática: **RESERVA PISTA DE PADEL** con el objetivo de que el examen de práctica puedan ser el mismo para todos los alumnos.
- Para intentar simular como se diseña y se produce software en el mundo real se deberá definir dos entornos (en el mundo real hay mucho más) uno de desarrollo y otro productivo. Además, esto facilitará las entregas porque se deberá tener en desarrollo la versión a entregar y en producción la última entrega realizada.

## Planificación de la Entrega

El proyecto constará de dos entregas: una relativa al frontend y otra relativa al backend:

Contenido	Proyecto de PAT en Equipo	Fecha de la Entrega
Backend	Backend en SprintBoot + Pruebas	14/02/2026
	Base de Datos + Pruebas	01/03/2026
Frontend	HTML + CSS + Pruebas	05/04/2026
	JavaScript + Pruebas + Funcionalidad completa	03/05/2026

Este proyecto se tiene que realizar mayoritariamente fuera de clase con soporte de los profesores para la resolución de dudas y preguntas que pueda surgir.

## Herramientas a utilizar

Para la realización de este proyecto y con el objetivo que sea lo más cercano a la realidad, no sólo se tendrá que documentar el diseño sino también realizar el desarrollarlo, pruebas y test de calidad del software para poder tener una visión completa de todas las etapas de la Ingeniería del Software. Para ello, y con el objetivo de unificar las herramientas a utilizar, a continuación, se detallan las herramientas que se utilizarán:

- Herramientas de desarrollo:
  - IntelliJ IDEA.- <https://www.jetbrains.com/idea/download/>
  - Visual Code.- <https://code.visualstudio.com/download>
  - JDK Versión 21
  - Repositorio de código.- <https://github.com/>
  - DevOps.- Utilizar la configuración CI/CD que ofrece GitHub Action y desplegado en <https://docs.render.com/>
  - Junit.- Realización de pruebas unitarias. <https://junit.org/junit5/>
- Herramienta de metodología ágiles para trabajo colaborativo:
  - Panel Kanban.- GitHub Projects

La entrega se realizará en Moodle en un apartado a tal efecto.

## Backend

En la primera entrega los grupos deberán aportar:

- Un documento en formato PDF con las secciones:
  - Explicaciones del proyecto.
  - Casos de Uso de la aplicación
  - Repositorio git en el que se encuentra el código.
  - Manual de usuario.
- Como el equipo estará formado por 4 personas se repartirán los roles:
  - Scrum Master
  - Program Manager
  - Analista
  - Release Manager/QA

- La página web será accesible desde GitHub Pages de GitHub.
- Se deberá integrar, al menos en la rama main, Integración Continua y Despliegue Continuo.

A continuación, se detalla los end-point a utilizar:

## Endpoints propuestos (API REST)

Método	Ruta	Descripción	Respuestas (mínimas esperadas)
POST	/pistaPadel/auth/register	Registrar usuario (rol por defecto USER).	<b>201</b> creado, <b>400</b> datos inválidos, <b>409</b> email ya existe
POST	/pistaPadel/auth/login	Login y obtención de token (o sesión).	<b>200</b> ok, <b>400</b> request inválida, <b>401</b> credenciales incorrectas
POST	/pistaPadel/auth/logout	Logout (invalidar sesión/token si aplica).	<b>204</b> ok, <b>401</b> no autenticado
GET	/pistaPadel/auth/me	Devuelve el usuario autenticado.	<b>200</b> ok, <b>401</b> no autenticado
Método	Ruta	Descripción	Respuestas (mínimas esperadas)
GET	/pistaPadel/users	(ADMIN) Listado de usuarios (paginable opcional).	<b>200</b> , <b>401</b> , <b>403</b> no autorizado
GET	/pistaPadel/users/{userId}	(ADMIN o dueño) Obtener un usuario por id.	<b>200</b> , <b>401</b> , <b>403</b> , <b>404</b> no existe
PATCH	/pistaPadel/users/{userId}	(ADMIN o dueño) Actualizar datos (nombre, teléfono...).	<b>200</b> , <b>400</b> , <b>401</b> , <b>403</b> , <b>404</b> , <b>409</b> (email duplicado)
Método	Ruta	Descripción	Respuestas (mínimas esperadas)
POST	/pistaPadel/courts	(ADMIN) Crear pista (nombre, ubicación, precio/hora, activa...).	<b>201</b> , <b>400</b> , <b>401</b> , <b>403</b> , <b>409</b> (nombre duplicado)
GET	/pistaPadel/courts	Listar pistas (filtro opcional active=true/false).	<b>200</b>
GET	/pistaPadel/courts/{courtId}	Obtener detalle de una pista.	<b>200</b> , <b>404</b>
PATCH	/pistaPadel/courts/{courtId}	(ADMIN) Modificar pista (precio, activa, etc.).	<b>200</b> , <b>400</b> , <b>401</b> , <b>403</b> , <b>404</b>
DELETE	/pistaPadel/courts/{courtId}	(ADMIN) Eliminar pista (o desactivar).	<b>204</b> , <b>401</b> , <b>403</b> , <b>404</b> , <b>409</b> (si hay reservas futuras, si queréis imponer regla)
Método	Ruta	Descripción	Respuestas (mínimas esperadas)
GET	/pistaPadel/availability	Consultar disponibilidad por <b>fecha</b> y opcionalmente pista. Query: date=YYYY-MM-DD&courtId=...	<b>200</b> , <b>400</b> (sin date / mal formato)
GET	/pistaPadel/courts/{courtId}/availability	Disponibilidad de una pista en una fecha. Query: date=YYYY-MM-DD	<b>200</b> , <b>400</b> , <b>404</b>
Método	Ruta	Descripción	Respuestas (mínimas esperadas)
POST	/pistaPadel/reservations	(USER) Crear reserva. Body típico: courtId, date,	<b>201</b> , <b>400</b> , <b>401</b> , <b>404</b> (pista no existe), <b>409</b> (slot ocupado)

		startTime, durationMinutes.	
GET	/pistaPadel/ reservations	(USER) Listar <b>mis reservas</b> . Filtros opcionales: from=...&to=...	200, 401
GET	/pistaPadel/ reservations/ {reservationId}	(USER dueño o ADMIN) Obtener una reserva.	200, 401, 403, 404
DELETE	/pistaPadel/ reservations/ {reservationId}	(USER dueño o ADMIN) Cancelar reserva.	204, 401, 403, 404, 409 (si no se puede cancelar por política)
PATCH	/pistaPadel/ reservations/ {reservationId}	(USER dueño o ADMIN) Reprogramar/cambiar reserva (opcional).	200, 400, 401, 403, 404, 409 (nuevo slot ocupado)
Método	Ruta	Descripción	Respuestas (mínimas esperadas)
GET	/pistaPadel/ admin/ reservations	(ADMIN) Ver reservas de todos (filtros: date, courtId, userId).	200, 401, 403
GET	/pistaPadel/ health	Healthcheck simple (útil para Render/CI).	200

## Código de Respuestas

- **409 Conflict** se usa para colisiones de negocio: *email duplicado, slot ya reservado, cancelación no permitida*.

tema-6

- **401 vs 403:**
  - **401** = no autenticado (no token/sesión).
  - **403** = autenticado pero sin permisos (p.ej. USER intentando crear pista).

## Modelo de Datos: Proyecto Reserva de Pistas de Pádel

### 1. Record: Usuario

Representa a cualquier persona que utiliza el sistema (jugador o administrador).

#### Campos

- **idUsuario**  
Identificador único del usuario.
- **nombre**  
Nombre del usuario.

- **apellidos**  
Apellidos del usuario.
- **email**  
Correo electrónico (único en el sistema).
- **password**  
Contraseña cifrada.
- **telefono**  
Teléfono de contacto.
- **rol**  
Rol del usuario en el sistema.  
Valores posibles: USER, ADMIN.
- **fechaRegistro**  
Fecha y hora de alta en el sistema.
- **activo**  
Indica si el usuario está activo o deshabilitado.

### Restricciones

- El email debe ser único.
- Un usuario puede tener **0..n reservas**.
- Solo los usuarios con rol ADMIN pueden gestionar pistas.

## 2. Record: Pista

Representa una pista de pádel disponible para reserva.

### Campos

- **idPista**  
Identificador único de la pista.
- **nombre**  
Nombre identificativo de la pista (ej. “Pista 1”).
- **ubicacion**  
Ubicación o descripción física.
- **precioHora**  
Precio de la pista por hora.
- **activa**  
Indica si la pista está disponible para reservas.
- **fechaAlta**  
Fecha de creación de la pista.

### Restricciones

- El nombre de la pista debe ser único.

- Una pista puede tener **0..n reservas**.
- No se puede reservar una pista inactiva.

### 3. Record: Reserva

Representa la reserva de una pista por parte de un usuario en una franja horaria.

#### Campos

- **idReserva**  
Identificador único de la reserva.
- **idUsuario**  
Usuario que realiza la reserva.
- **idPista**  
Pista reservada.
- **fechaReserva**  
Fecha de la reserva (día).
- **horaInicio**  
Hora de inicio de la reserva.
- **duracionMinutos**  
Duración de la reserva en minutos.
- **horaFin**  
Hora de finalización (calculada).
- **estado**  
Estado de la reserva.  
Valores posibles: **ACTIVA, CANCELADA.**
- **fechaCreacion**  
Fecha y hora en la que se realiza la reserva.

#### Restricciones

- No pueden existir dos reservas solapadas para la misma pista.
- Un usuario solo puede modificar o cancelar sus propias reservas.
- Una reserva pertenece **exactamente a un usuario y a una pista**.

### 4. Record: Disponibilidad (conceptual)

No necesariamente debe almacenarse en base de datos.  
Se puede calcular a partir de las reservas existentes.

#### Campos conceptuales

- **idPista**
- **fecha**
- **franjaDisponibles**  
Lista de tramos horarios libres.

## Regla

- La disponibilidad se calcula en función de:
  - Horario del club (definido por la aplicación).
  - Reservas existentes para esa pista y fecha.

## 5. Record: Rol

Puede ser tabla o enumeración.

## Campos

- **idRol**
- **nombreRol** (USER, ADMIN)
- **descripcion**

## Relaciones entre records

- **Usuario 1 — N Reserva**
- **Pista 1 — N Reserva**
- **Reserva N — 1 Usuario**
- **Reserva N — 1 Pista**
- **Usuario N — 1 Rol**

## Trazas

- La aplicación debe introducir trazas de nivel debug, error e info para la trazabilidad de la misma.

## Tareas programadas

- Todas las noches a las 2 de la mañana deberá mandar un correo a los usuarios que tienen pista reservada ese día como recordatorio.
- El primer día del mes se mandará un correo a todos los usuarios con las pistas y los horarios disponibles.

## Pruebas

- Se debe introducir diferentes tipologías de pruebas cubiertas en al asignatura. .

## Frontend

En la primera entrega los grupos deberán aportar:

- Un documento en formato PDF con las secciones:
  - Explicaciones del proyecto.
  - Casos de Uso de la aplicación
  - Repositorio git en el que se encuentra el código.
  - Manual de usuario.
- Como el equipo estará formado por 4 personas se repartirán los roles:
  - Scrum Master
  - Program Manager
  - Analista
  - Release Manager/QA
- La página web será accesible desde GitHub Pages de GitHub.
- Se deberá tener una rama principal en el repositorio *main*, los desarrollos de los equipos se deberán realizar en una rama *develop*.
- Se deberá integrar, al menos en la rama *main*, Integración Continua y Despliegue Continuo.

**La evaluación de la práctica se realizará vía examen en dos pruebas que cada una tendrá un peso del 50% respecto la nota final de la práctica.**