MEMORIA FINAL DEL PROYECTO

SPORTIFY

Tu diario deportivo digital

Alumno: Manuel Gómez Martín

ÍNDICE

1. Introducción

Sportify nace de la necesidad de una plataforma digital que combine el periodismo deportivo con la participación activa del usuario. A través de funcionalidades como comentarios, likes, filtrado personalizado de contenido y sistema de roles, se ofrece una experiencia única tanto para consumidores como creadores de contenido deportivo.

2. Plan de Empresa

JUSTIFICACIÓN

- ¿Por qué este negocio?
  A raíz del gusto por el deporte, aparece la idea de crear un diario Deportivo online e interactivo, para que todo el que quiera pueda compartir opiniones y diferentes noticias sobre cualquier tipo de deporte.
- ¿Existe competencia?
  Sí, medios como Marca, AS, ESPN o apps como OneFootball. Lo que nos diferencia del resto es la interacción de cualquier usuario que quiera ser partícipe de este proyecto.
- ¿Por qué creo que se venderá?
  Porque mezcla lo mejor del periodismo deportivo con lo mejor de las redes sociales.
- ¿Qué me diferencia?
  Personalización, roles, interacción social y participación.
- ¿Qué hay de novedoso?
  Sistema híbrido entre red social y diario, tecnología moderna y abierta.

NOMBRE y LOGO

- Nombre: SPORTIFY. Inspirado en 'Sport' y 'Spotify'. Transmite dinamismo y personalización.
- Justificación del logo: Circular, con tipografía moderna e inspiración en Spotify, pero con identidad deportiva clara.

PRODUCTO

- Lectura de noticias deportivas, interacción con comentarios y likes, gestión de cuenta y roles.
- Captura de interfaz o diseño (ver prototipo en el apartado 4.1).
- Necesidad cubierta: actualidad deportiva personalizada con participación.

MODELO DE NEGOCIO

- Freemium, publicidad, afiliación, suscripciones sin anuncios.

CONSUMIDORES Y POSIBLES CLIENTES

- Cualquier amante del deporte que quiera estar informado sobre todo lo relacionado con su deporte o equipo favorito.
- Zona inicial: España. Escalabilidad global.

COMPETENCIA

- Marca/AS/ESPN/OneFootball: ofrecen noticias pero sin interacción personalizada ni creación de contenido.
- Sportify: cubre ese hueco.

DAFO

- Debilidades: dependencia técnica, necesidad de contenido.
- Amenazas: copia del modelo por grandes empresas.
- Fortalezas: plataforma terminada, moderna y funcional.
- Oportunidades: nicho sin explotar, crecimiento como red temática.

PUBLICIDAD Y PROMOCIÓN

- Soportes: TikTok, Instagram, Google Ads, YouTube Shorts.
- Coste: inversión inicial de 300 € en redes sociales e influencers.

3. Descripción detallada del sistema y listado de historias de usuario

LISTADO DE HISTORIA DE USUARIOS:

Title URL

HU-01 Conexion con postgres https://github.com/manuelgomez04/Sportify/issues/1

HU-39 DockerFile https://github.com/manuelgomez04/Sportify/issues/79

HU-35 Implementar seguridad https://github.com/manuelgomez04/Sportify/issues/35

HU-36 Documentar los métodoshttps://github.com/manuelgomez04/Sportify/issues/45

HU-38 Filtrados https://github.com/manuelgomez04/Sportify/issues/64

HU-04 Editar usuario https://github.com/manuelgomez04/Sportify/issues/4

HU-41 Página Home https://github.com/manuelgomez04/Sportify/issues/82

HU-02 Registrar usuario https://github.com/manuelgomez04/Sportify/issues/2

HU-03 Iniciar Sesión https://github.com/manuelgomez04/Sportify/issues/3

HU-40 Login Angular https://github.com/manuelgomez04/Sportify/issues/81

HU-05 Eliminar usuario https://github.com/manuelgomez04/Sportify/issues/5

HU-30 Mostrar favoritos https://github.com/manuelgomez04/Sportify/issues/30

HU-32 Deportes favoritos https://github.com/manuelgomez04/Sportify/issues/32

HU-31 Equipos Favoritos https://github.com/manuelgomez04/Sportify/issues/31

HU-33 Ver likes https://github.com/manuelgomez04/Sportify/issues/33

HU-34 Ver comentarios https://github.com/manuelgomez04/Sportify/issues/34

HU-43-Añadir foto de perfil https://github.com/manuelgomez04/Sportify/issues/95

HU-06 Escribir noticia https://github.com/manuelgomez04/Sportify/issues/6

HU-27 Añadir deporte a noticia https://github.com/manuelgomez04/Sportify/issues/27

HU-28 Añadir liga a noticia https://github.com/manuelgomez04/Sportify/issues/28

HU-29 Añadir un equipo a una noticia https://github.com/manuelgomez04/Sportify/issues/29

HU-07 Editar una noticia https://github.com/manuelgomez04/Sportify/issues/7

HU-18 Crear equipo https://github.com/manuelgomez04/Sportify/issues/18

HU-37 Subida de archivos https://github.com/manuelgomez04/Sportify/issues/49

HU-42 Documentación https://github.com/manuelgomez04/Sportify/issues/83

HU-08 Borrar noticia https://github.com/manuelgomez04/Sportify/issues/8

HU-09 Escribir un comentario https://github.com/manuelgomez04/Sportify/issues/9

HU-10 Editar un comentario https://github.com/manuelgomez04/Sportify/issues/10

HU-11 Eliminar un comentario https://github.com/manuelgomez04/Sportify/issues/11

HU-25 Seguir deporte https://github.com/manuelgomez04/Sportify/issues/25

HU-12 Dar me gusta https://github.com/manuelgomez04/Sportify/issues/12

HU-13 Quitar me gusta https://github.com/manuelgomez04/Sportify/issues/13

HU-22 Dejar de seguir equipo https://github.com/manuelgomez04/Sportify/issues/22

HU-26 Dejar de seguir deporte https://github.com/manuelgomez04/Sportify/issues/26

HU-21 Seguir a un equipo https://github.com/manuelgomez04/Sportify/issues/21

HU-23 Seguir liga https://github.com/manuelgomez04/Sportify/issues/23

HU-24 Dejar de seguir liga https://github.com/manuelgomez04/Sportify/issues/24

HU-14 Añadir un deporte https://github.com/manuelgomez04/Sportify/issues/14

HU-15 Eliminar un deporte https://github.com/manuelgomez04/Sportify/issues/15

HU-16 Crear una liga https://github.com/manuelgomez04/Sportify/issues/16

HU-17 Eliminar una liga https://github.com/manuelgomez04/Sportify/issues/17

HU-44 Tablas de admin https://github.com/manuelgomez04/Sportify/issues/110

HU-19 Eliminar un equipo https://github.com/manuelgomez04/Sportify/issues/19

HU-45 Cambios visuales https://github.com/manuelgomez04/Sportify/issues/115

DESCRIPCIÓN DETALLADA DEL SISTEMA:
El programa se ejecuta en el puerto Localhost:8080

La documentación está realizada con swagger

Realiza diferentes tipos de operaciones con Noticias, Deportes, Ligas, Usuarios y Equipos

A la hora de ejecutar el programa habrá que escribir en la terminal el comando docker-compose up --build

Para poder arrancar el programa necesitas poner en el application-mail.properties tu correo y tu contraseña para que pueda compilar el programa

Funcionalidades

En función del tipo de usuario con el que inicies sesión

Registro: Los usuarios pueden registrarse como regulares, escritores o administradores. Inicio de sesión: Los usuarios pueden iniciar sesión y obtener un token JWT. Editar perfil: Los usuarios pueden actualizar su información personal. Eliminar cuenta: Los usuarios pueden eliminar su cuenta. Noticias

Crear noticia: Los escritores pueden publicar nuevas noticias. Obtener noticias: Los usuarios pueden ver una lista paginada de noticias. Editar noticia: Los escritores pueden modificar sus noticias. Eliminar noticia: Los escritores pueden eliminar sus noticias. Filtrar noticias: Los usuarios pueden filtrar noticias por el titular, una fecha de creación entre dos rangos, por el deporte, la liga y. Deportes

Crear deporte: Los administradores pueden agregar nuevos deportes. Eliminar deporte: Los administradores pueden eliminar deportes. Seguir deporte: Los usuarios pueden seguir deportes de su interés. Dejar de seguir deporte: Los usuarios pueden dejar de seguir deportes. Ligas

Crear liga: Los administradores pueden crear nuevas ligas asociadas a un deporte. Seguir liga: Los usuarios pueden seguir ligas. Dejar de seguir liga: Los usuarios pueden dejar de seguir ligas. Equipos

Crear equipo: Los administradores pueden agregar nuevos equipos. Seguir equipo: Los usuarios pueden seguir equipos. Dejar de seguir equipo: Los usuarios pueden dejar de seguir equipos. Comentarios

Agregar comentario: Los usuarios pueden comentar en las noticias. Editar comentario: Los usuarios pueden editar sus comentarios. Eliminar comentario: Los usuarios pueden eliminar sus comentarios, y los administradores pueden eliminar cualquier comentario. Likes Dar like: Los usuarios pueden dar like a las noticias. Eliminar like: Los usuarios pueden quitar su like de una noticia.

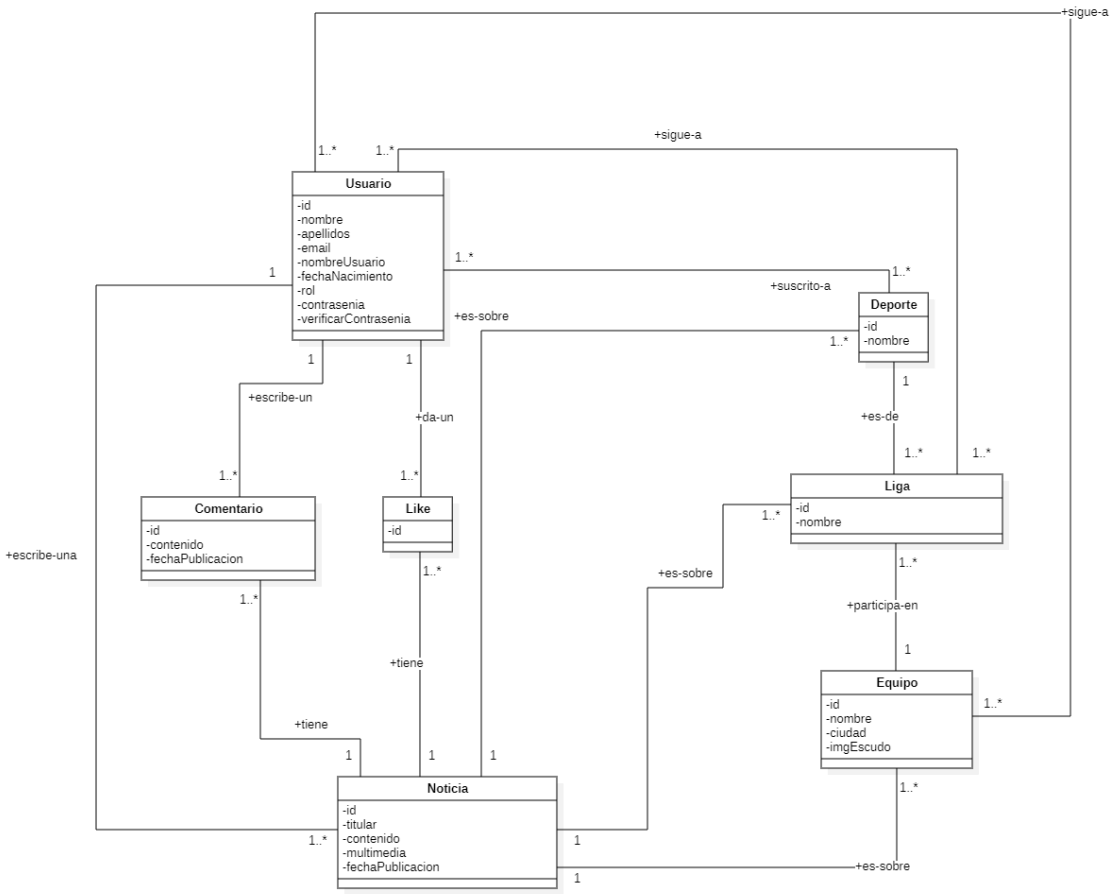
4. Modelado y diseño

4.1 Prototipado en Figma de la aplicación

Se ha diseñado la interfaz en Figma representando la experiencia de usuario tanto en versión web como móvil.

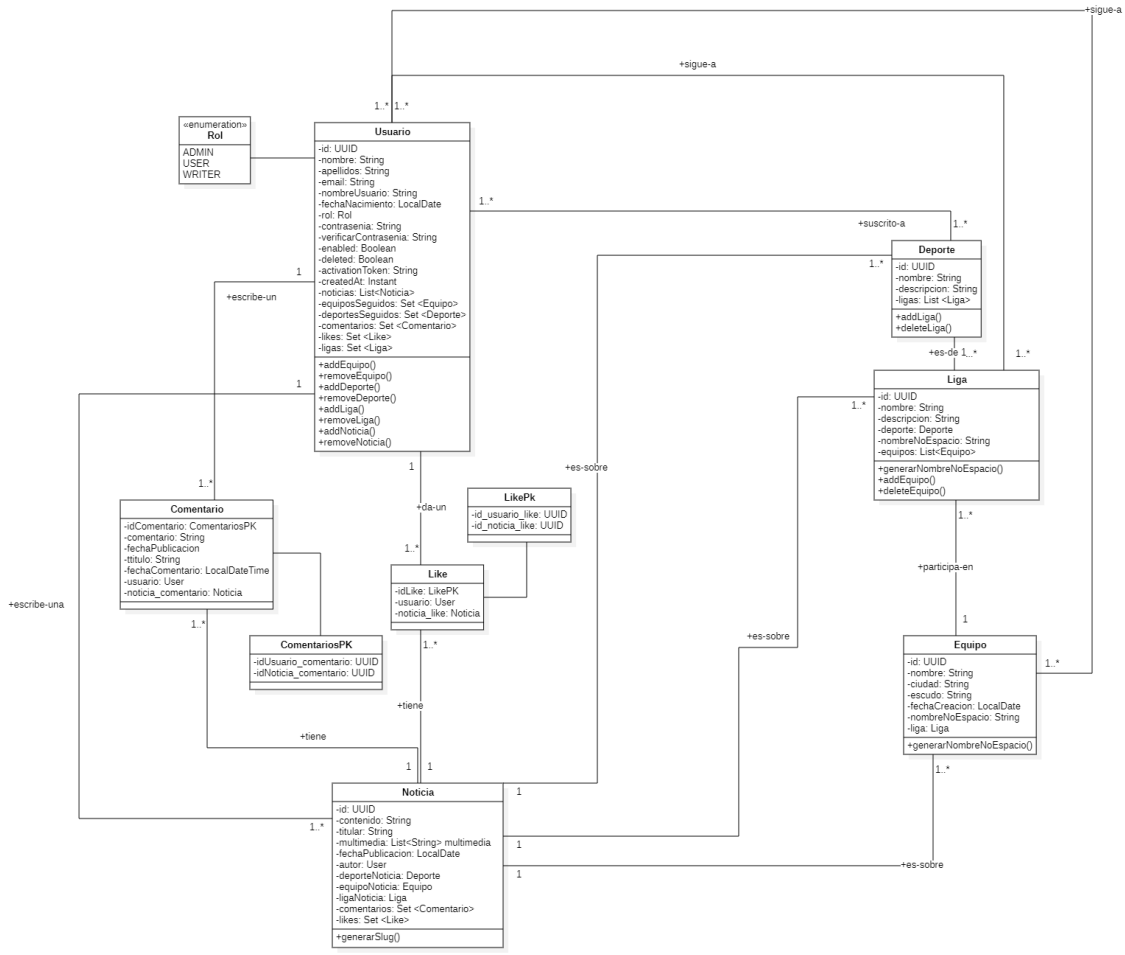
https://www.figma.com/design/HknyihjQ9WE5PwS4aCbiAc/SportifyWeb?node-id=38-1452&t=WgpIuGqkLTSESCfn-1

4.2 Diagrama de clases del modelo de dominio de la API



5. Diseño

5.1 Diagrama de clases del diseño indicando: direccionalidad de las asociaciones, tipos y los tipos de datos a utilizar.



6. Implementación

6.1 Descripción de los diferentes paquetes y clases de cada aplicación.

SPORTIFY

Paquete principal:

- com.salesianos.dam.sportify

Contiene la clase principal de arranque de la aplicación Spring Boot.

SportifyApplication:

- Clase con el método main que inicia la aplicación.

Paquete comentario:

Gestiona los comentarios de las noticias.

controller/ComentarioController:

- Controlador REST para crear, editar, borrar y listar comentarios.

service/ComentarioService:

- Lógica de negocio para gestionar comentarios y sus relaciones con usuarios y noticias.

model/Comentario, ComentariosPk:

- Entidades que representan los comentarios y su clave primaria compuesta.

dto/:

- Clases para la transferencia de datos de comentarios.

Paquete deporte:

Gestiona los deportes y sus relaciones.

controller/DeporteController:

- Controlador REST para crear, borrar y listar deportes.

service/DeporteService:

- Lógica de negocio para la gestión de deportes y la eliminación en cascada.

model/Deporte:

- Entidad que representa un deporte.

dto/:

- Clases para la transferencia de datos de deportes.

Paquete equipo:

Gestiona los equipos deportivos.

controller/EquipoController:

- Controlador REST para crear, borrar y listar equipos, y obtener equipos por liga.

service/EquipoService:

- Lógica de negocio para la gestión de equipos y su relación con usuarios y noticias.

model/Equipo:

- Entidad que representa un equipo.

dto/:

- Clases para la transferencia de datos de equipos.

Paquete liga:

Gestiona las ligas deportivas.

controller/LigaController:

- Controlador REST para crear, borrar y listar ligas, y obtener ligas por deporte.

service/LigaService:

- Lógica de negocio para la gestión de ligas y su relación con usuarios, equipos y noticias.

model/Liga:

- Entidad que representa una liga.

dto/:

- Clases para la transferencia de datos de ligas.

Paquete noticia:

Gestiona las noticias deportivas.

controller/NoticiaController:

- Controlador REST para crear, editar, borrar y listar noticias, y asociarlas a equipos, ligas o deportes.

service/NoticiaService:

- Lógica de negocio para la gestión de noticias y su relación con usuarios, equipos, ligas y deportes.

model/Noticia:

- Entidad que representa una noticia.

dto/:

- Clases para la transferencia de datos de noticias.

Paquete user:

Gestiona los usuarios y su autenticación.

controller/UserController:

- Controlador REST para registro, edición, borrado, login/logout y gestión de favoritos de los usuarios.

service/UserService:

- Lógica de negocio para la gestión de usuarios, edición, borrado, seguimiento y roles.

model/User, Role:

- Entidades que representan a los usuarios y sus roles.

dto/:

- Clases para la transferencia de datos de usuarios.

Otros paquetes relevantes

files:

- Gestión de archivos (imágenes, multimedia) asociados a entidades.

error:

- Excepciones personalizadas para el manejo de errores.

util:

- Utilidades varias, como servicios de email o criterios de búsqueda.

Resumen

Cada paquete separa controladores (API REST), servicios (lógica de negocio), modelos (entidades JPA) y DTOs (transferencia de datos). La lógica de desvinculación y borrado en cascada mantiene la integridad de las relaciones entre usuarios, equipos, ligas, deportes y noticias.

SPORTIFYANGULAR

Carpeta principal:

- src/app/

Contiene todo el código fuente de la aplicación Angular.

app.module.ts:
- Módulo principal de la aplicación. Declara y agrupa todos los componentes, servicios y módulos necesarios.

app-routing.module.ts:
- Define las rutas de la aplicación, asociando URLs a componentes concretos.

app.component.ts / app.component.html / app.component.css:
- Componente raíz de la aplicación. Sirve como punto de entrada visual y lógica principal.

Carpeta components/
- Aquí se agrupan los diferentes componentes reutilizables y páginas de la aplicación.

Componentes de páginas:
home.component.ts, login.component.ts, register.component.ts, noticias.component.ts, etc.

Cada uno representa una vista o página principal de la aplicación (inicio, login, registro, listado de noticias, detalle de noticia, etc).

Componentes reutilizables

navbar.component.ts
Es una interfaz que se usa en varias páginas.

Carpeta models/:
- Contiene las interfaces y clases TypeScript que definen la estructura de los datos usados en la aplicación. noticia.model.ts, usuario.model.ts, comentario.model.ts, etc.

Carpeta services/:
- Aquí se encuentran los servicios que gestionan la lógica de negocio y la comunicación con el backend. Algunos ejemplos son:

noticia.service.ts: Métodos para obtener, crear, editar y borrar noticias.

usuario.service.ts: Métodos para login, registro y gestión de usuarios.

comentario.service.ts: Métodos para gestionar comentarios.

Carpeta guards/:
- Contiene los guardias de rutas, que controlan el acceso a ciertas páginas según el estado de autenticación o el rol del usuario. auth.guard.ts, admin.guard.ts.

Carpeta interceptors/:

-Incluye interceptores HTTP para añadir cabeceras (como el token JWT) a las peticiones o gestionar errores globalmente.
Ejemplo: auth.interceptor.ts.

Carpeta assets/:
- Archivos estáticos como imágenes, iconos, etc.

Carpeta environments/:
- Archivos de configuración de entorno (environment.ts, environment.prod.ts) para definir variables como la URL base de la API según el entorno (desarrollo o producción).

Otros archivos relevantes:

proxy.conf.json
Archivo de configuración del proxy para redirigir las peticiones API al backend durante el desarrollo o en Docker.

styles.css
Estilos globales de la aplicación.

Resumen:
La aplicación Angular está organizada en módulos, componentes, servicios, modelos y utilidades. Cada carpeta agrupa elementos según su función: vistas, lógica de negocio, modelos de datos, protección de rutas, interceptores y recursos estáticos. La comunicación con el backend se realiza a través de servicios, y el enrutamiento permite la navegación entre las diferentes páginas de la aplicación.