IES SAN JUAN DE LA CRUZ
Consejería de Educación, Ciencias
y Universidades
Comunidad de Madrid

UNIÓN EUROPEA
Fondo Social Europeo
"El FSE invierte en tu futuro"

TÉCNICO SUPERIOR EN DESARROLLO DE APLICACIONES WEB (IFCS03)

MÓDULO: PROYECTO. CURSO: 2023-2024
CONVOCATORIA: EXTRAORDINARIA

Título del proyecto: Fototren.es

PROFESOR-COORDINADOR: Jorge Carlón Barea
ALUMNO:
APELLIDOS Y NOMBRE: Santiago Alvarez Hernandez DNI: 49046444W

MODALIDAD DE PROYECTO:
MODELO A): Proyecto de investigación experimental.

Índice

1. Introducción

1.1 Descripcion y justificacion del proyecto

FotoTren.es es una aplicación web que utiliza Angular como framework principal para el desarrollo del entorno del cliente, para apartado visual FotoTren.es utiliza Angular Material que es un módulo desarrollado para su uso en Angular. En cuanto al apartado del BackEnd FotoTren.es utiliza node.js para servir los documentos, filtrar los datos y guardarlos en la base de datos.

La mayor parte de la aplicación está desarrollada para que se asemeje lo máximo posible a la realidad, los datos no son filtrados por el cliente, son filtrados por el servidor Node.js, las imágenes no se obtienen directamente leyendo la carpeta de trabajo de la aplicación que lo esté desplegando, si no que para acceder a los datos tienes que pedirlos a un servidor que el hará los filtros necesarios para asegurar la integridad de los datos.

En cuanto al apartado de su uso es un foro de internet con dos maneras de interactuar con otros usuarios:

- Subir fotografias para que otras personas las puedan ver
- Entrar en el foro y responder a los hilos principales, o crear un hilo tú mismo

1.2 Objetivos del proyecto

Objetivo principal

Crear un espacio de interacción entre usuarios amantes del ferrocarril español sin la necesidad del uso de plataformas de terceros, o el uso de páginas web creadas a inicios de los 2000. Que sea rápida y segura.

Objetivos secundarios

Que la experiencia de usuario sea lo mas simple posible para atraer a una gran cantidad de usuarios. Que las bases de datos permitan en un futuro la ampliacion de caracteristicas y que sean compatibles con las caracteristicas anteriores

Competencias adquiridas

Durante el desarrollo he adquirido conocimientos en Node.js, Angular, Angular Material y arquitectura hexagonal de proyectos.

2. Metodología

2.1 Planificación del proyecto

Semana	Objetivo	Tarea
1-2	Sentar las bases backend	- Crear modelo ER de la BBDD en MySQL - Seleccionar lenguaje servidor
3-4	Creación BBDD	- Crear la BBDD con todos los requisitos de la semana anterior - Iniciar la conexión entre Servidor y BBDD
5-6	Creación Login	- Acceder a la BBDD pedir datos de Inicio de sesión y devolver si puede iniciar sesión - Creación cuenta de usuario
7-8	Creación interfaz de usuario	- Creación de los estilos bases que usará toda la web
9-10	Creación de forms	- Creación de la página para ingresar nuevas fotos - Creación de la página del foro de discusiones
11-12	Creacion paginas faltantes	- Creación página para mostrar las fotos ya subidas - Creación de la pagina para ver los hilos de discusiones
13-14	Testing final	- Optimización de las paginas - Realización de pruebas - Realización de correcciones de errores

3. Tecnologías utilizadas

Draw.io: He usado esta aplicación para el diseño conceptual de la BBDD.

LAMPP: He utilizado LAMPP (la versión de linux de XAMPP) para alojar la base de datos de MySQL, permite la creación de BBDD de manera rápida y usando contenedores, por lo que no es un MySQL normal, es una version light del mismo.

MySQL Workbench: He usado esta herramienta para el diseño de la BBDD y para interactuar con la BBDD en un entorno gráfico.

NeoVim: Ha sido el IDE que he usado para escribir código, es un IDE de manejo por teclado.

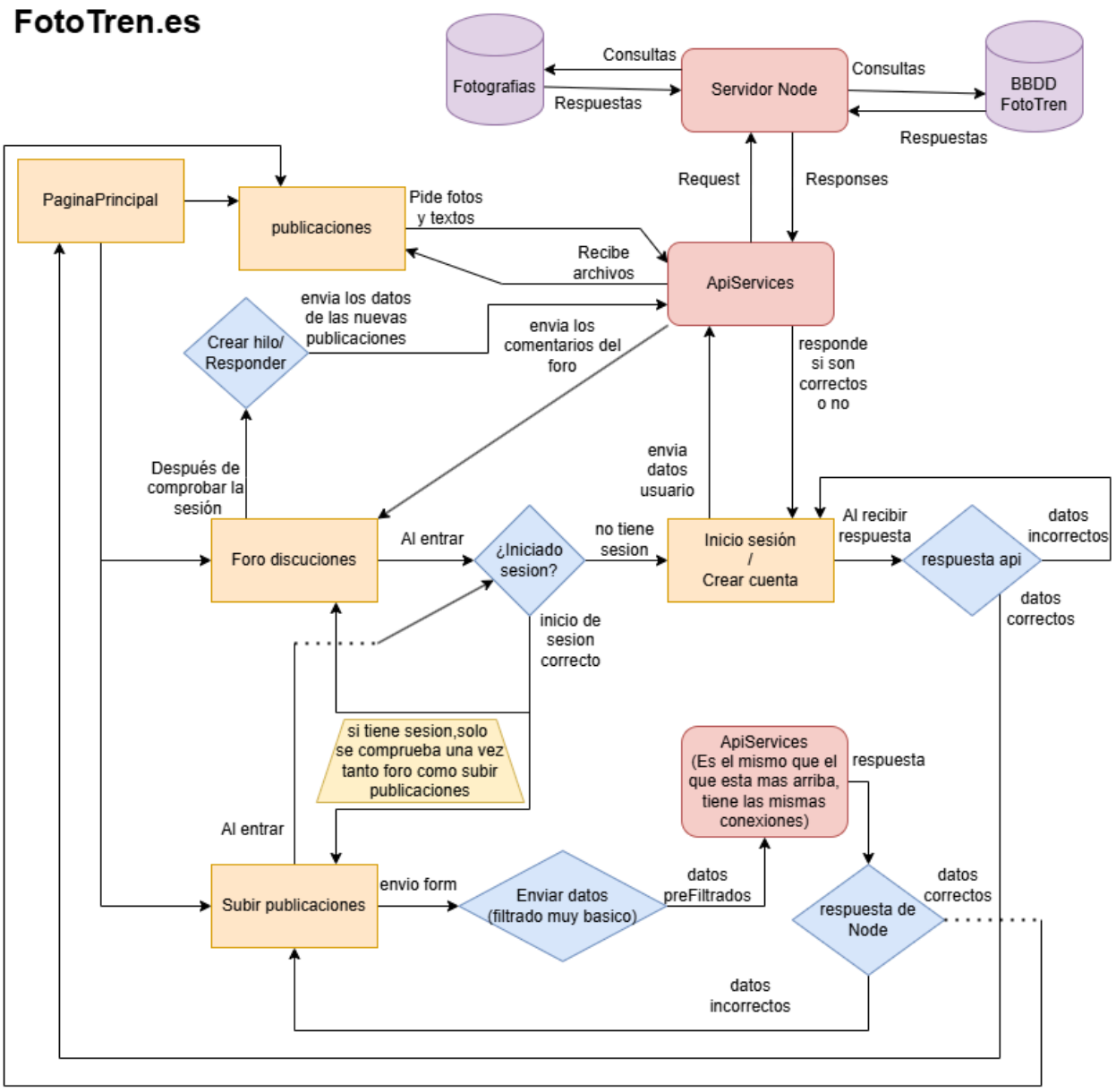
Node.js: Es un lenguaje basado en JavaScript orientado al entorno del servidor, de código abierto, con arquitectura orientada a eventos y basado en el motor V8 de google.

Angular: Es un framework orientado al desarrollo de aplicaciones web, que usa TypeScript, de código abierto mantenido por Google.

Angular Material: Es un módulo de Angular que permite implementar componentes Angular con un diseño basado en Material Design.

4. Diseño de la aplicación web

4.1 Diagrama de flujo de la aplicación



Leyenda

Las formas y colores indican que es cada apartado.

Las flechas indica que hay un flujo de datos, la descripción de que lleva cada dato o condición de salida lo dará el campo de texto mas cercano a el punto donde nace, es decir donde no esta la flecha

en este caso lleva un 2 y no un 1

texto — Paginas

texto — Aclaraciones

texto — Acciones o funciones

Flujo de datos o archivos

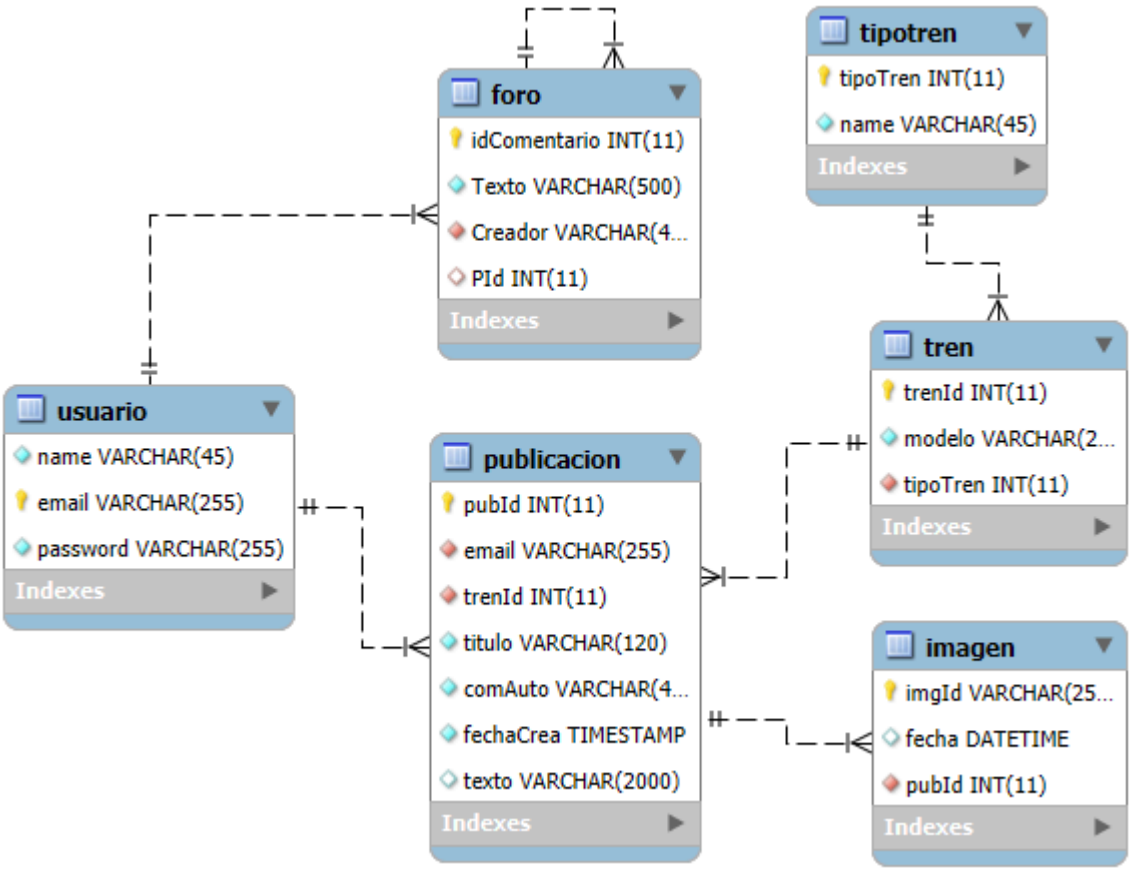
texto — Servicios o servidores que condensan Funciones en el backend

texto — Base de datos

1 ← 2

Hecho por Santiago Álvarez Hernández

4.2 Modelos relacionales de la BBDD



4.3 Diseño y análisis de la BBDD

I. TipoTren:
   A. tipoTren: Es un número que gracias a un AutoIncrement permite guardar solamente una vez el tipo de tren que es en una tabla y que la tabla tren solo guarde un int y no un varchar. Es la PK.
   B. name: Es el nombre al que querría hacer referencia, por ejemplo Alvia, Media Distancia, AVE, etc.

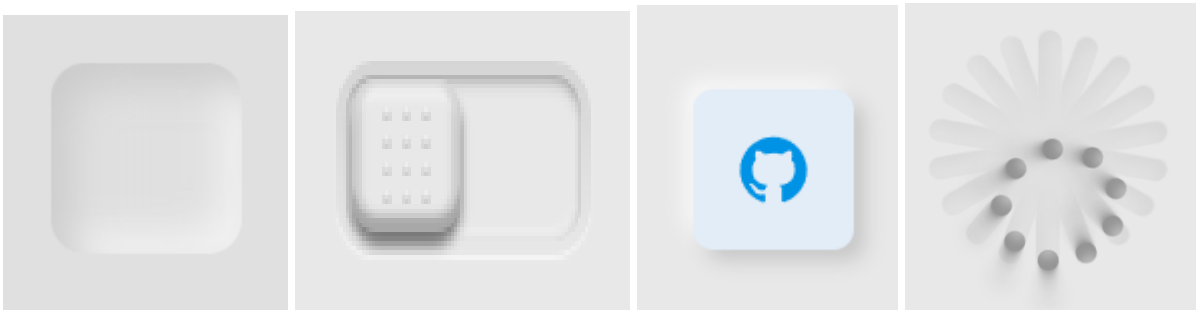
II. Tren:
   A. TrenId: Es el id del tren que acabamos de subir, se diferencia de tipo tren ya que aqui hablamos de modelos y no de servicios. Es la PK.
   B. Modelo: Es el nombre por el que conocemos a un tren, un Alvia puede ser varios modelos, un Renfe s-130, un Renfe s-730, un Renfe s-252 + Ramas de Talgo 7, etc.
   C. tipoTren: Es una Foreign Key que hace referencia a la tabla tipoTren, a su columna tipoTren.

III. Usuario:
   A. Email: Es el correo electrónico del usuario, es la Primary key.
   B. Name: Es el nombre del usuario, es unique al igual que email aunque no sea la pk de la tabla
   C. Password: Es un campo que guarda las contraseñas encriptadas

IV. Foro:
    A. idComentario: Es la primary key, es un número que va auto incrementando de uno en uno, sirve para localizar los comentarios.
    B. Texto: Es el texto que lleva la respuesta o el hilo.
    C. Creador: Es una FK que relaciona con la tabla usuario en la columna name. Por eso name es unique para poder permitir estas operaciones sean más simples y no haya que hacer una consulta conjunta para sacar el comentario y el creador
    D. PId: Es el id de comentario del hilo del que desprende, es una auto FK, ya que hace referencia a la misma tabla solo que apunta a la columna idComentario, puede ser null, ya que si no tiene PId es un hilo principal
V. Publicación:
    A. PubId: Es el identificador de la publicación, es un int con autoincrement y además es la PK.
    B. Email: Relaciona una publicación a un usuario, es una fk que relaciona la tabla usuario, la columna email.
    C. TrenId: Relaciona el tren en específico con la publicación
    D. Título: Es el título de la publicación
    E. ComAuto: Es la localización de la foto.
    F. fechaCrea: Es la fecha en la que se subió la publicación, tiene un valor por defecto de la hora de la consulta de insert.
    G. Texto: Es el texto que tiene la publicación, no es el título, es distinto.
VI. Imagen
    A. ImgId: El identificador de la imagen es el nombre del archivo, que contiene el nombre original y una timestamp de la fecha de subida para evitar duplicados. Es la Primary Key.
    B. Fecha: Es redundante pero está puesta para ver si están habiendo problemas de latencia entre cuando lo procesa ServerService Api y cuando llega a insertarse.
    C. PubId: Es una FK que relaciona una imagen con la publicación a la que corresponde

5. Diseño de la página

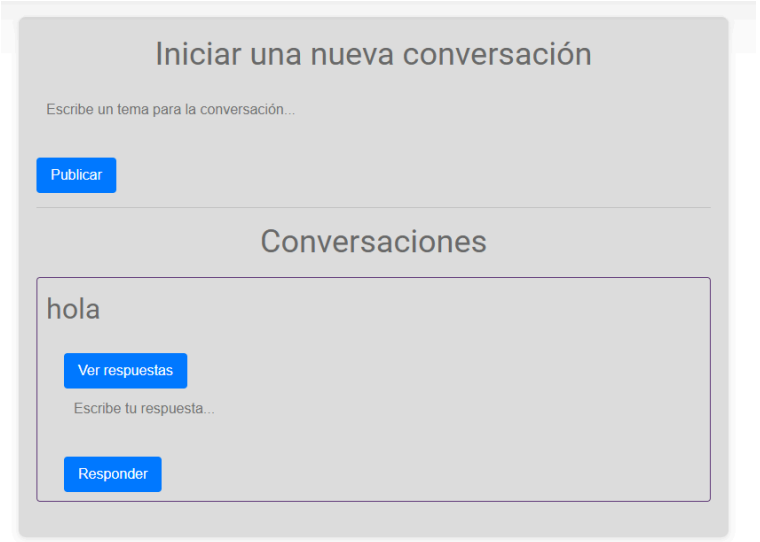
Es una pagina que sobretodo usa Angular material y utiliza un estilo de diseño web llamado ‘Neomorfismo’ (Neumorphism), que se basa en generar un supuesto relieve para distinguir apartados gráficos sin necesidad de usar colores de paletas opuestas, algunos ejemplos son:



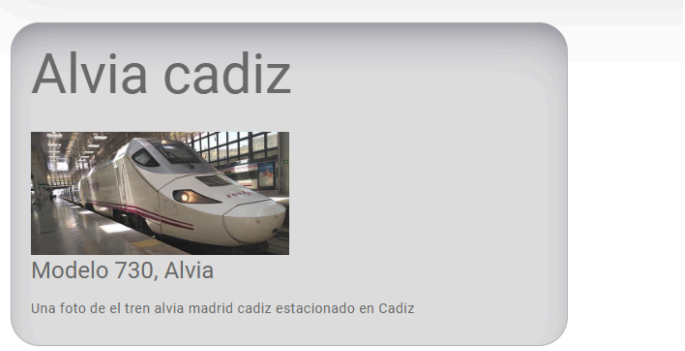
Para esto me he valido sobre todo de la función ‘box-shadow’ y ‘linear-gradient’ de CSS Sobretodo usando colores muy grisáceos

La aplicación web está dividida en Varias paginas

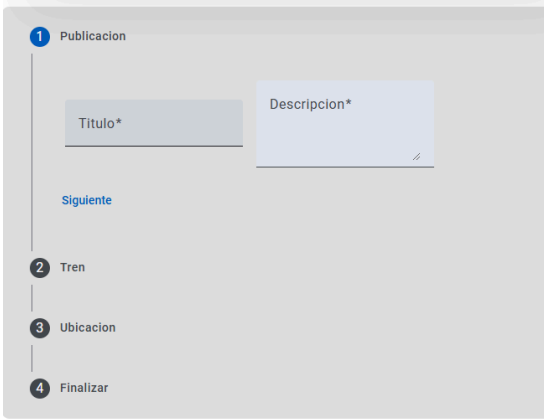
Foro:



Publicaciones:



Nuevas publicaciones:



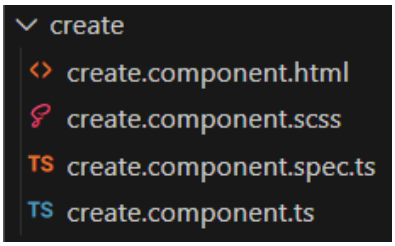
Y las dos pantallas de Usuarios

6. Implementación

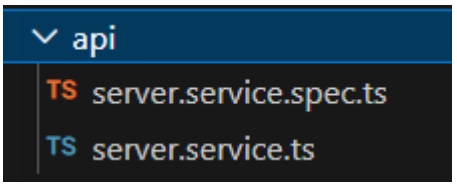
6.1 Arquitectura

Al usar Angular la arquitectura Modelo-Vista-Controlador es muy sencilla de implementar:
- Los modelos son gestionados por los Servicios que Angular te facilita, en mi caso tengo 4 servicios que se encargan de la comunicación con el Servidor Node además de encargarse de transmitir datos entre componentes. Todos se encuentran en src/app/services
- Los controladores son los ts que contienen cada componente ya que le dan la lógica de ese componente y gracias a servicios pueden hablarse entre ellos
- Las vistas son gestionadas por los HTML que tiene cada component. Los componentes son las unidades básicas de una página web, cuenta cada componente con 4 archivos,
    - HTML, es un html que puede alterar Angular en cualquier momento, puede omitirse y simplemente poner el HTML dentro del TS
    - CSS/SCSS/etc. Contiene la hoja de estilos que se va a aplicar sobre el html del componente, se puede omitir o se puede escribir directamente en el TS al igual que sucedía con el HTML
    - Spec.TS contiene código que asegura la intercompatibilidad entre componente, no se debe tocar, es autogenerado.
    - TS contiene la lógica que tendrá ese component, pueden comunicarse entre ellos sin necesidad de servicios pero implica el uso de propagación de eventos que si no se usan correctamente pueden llevar a una gran bajada de rendimiento para el usuario o para el servidor

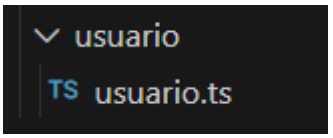
Este es el ejemplo de como se ve un component



Este es un ejemplo de como se ve un servicio



Así se ve una clase



7. Conclusión

7.1 Dificultades

El uso de Angular en vez de un lenguaje que conociera, sumado a la dificultad de programar en un nivel de abstracción tan alto como es el programar con componentes, servicios y sobretodo la dificultad de entender el proceso de Hidratación de Angular y como usarlo para mi beneficio, han sido las mayores dificultades que he ido encontrando por el camino.

Otras dificultades fueron el usar Node.js como BackEnd ya que usa JavaScript que no tiene nada que ver con Java. Si pudiera volver a empezar seguramente usaría Java con sus Servlets o C# con Core y .NET.

7.2 Apartados a mejorar

El apartado visual es mi punto flaco, me gustaría ser capaz de hacer un proyecto no solo que el apartado del backend sea bonito, si no que el frontend también lo sea.

Una implementación más compartimentada de

8. Bibliografía

1. Documentacion de angular https://angular.dev/tutorials
2. Documentacion de Angular Material https://material.angular.io/components/categories
3. Servidor Node orientado a api rest (El paso de microsoft sql a mysql lo hice yo) https://youtu.be/VuQAF-44Lo0
4. Generador de elementos neomorfistas https://neumorphism.io/#e0e0e0
5. Tutoriales angular desde 0 https://www.youtube.com/watch?v=w5xhZZx031I&list=PL42UNLc8e48RINrNGumxAKulG5CWgs_yv