Documentación Técnica – Aplicación Web Rally Fotográfico

ExpressCaptureVz®

Proyecto Integrado Final
Desarrollo de Aplicaciones Web
Angela Borges Cantarino

Índice

1. Justificación técnica de las tecnologías utilizadas

Fronted

Angular (Standalone)

Se eligió Angular como framework de frontend por ser una herramienta moderna, robusta y ampliamente utilizada en el desarrollo de aplicaciones web de una sola página (SPA). Además la versión standalone permite una mayor modularidad y simplificación del código, sin necesidad de módulos tradicionales. Angular ofrece:

- **Componentización** clara del proyecto.
- **Data binding** eficiente.
- **Validaciones de formularios** integradas.
- Excelente integración con herramientas modernas como Bootstrap y FontAwesome, etc...

Bootstrap

Utilicé Bootstrap para facilitar un diseño responsivo y profesional sin necesidad de diseñar CSS desde cero, aunque luego haya personalizado mucho código para mi necesidad. Su sistema de rejilla, componentes reutilizables y compatibilidad con Angular ayudan a construir una interfaz clara y funcional.

FontAwesome

Se incorporó FontAwesome para dotar a la aplicación de iconos visuales que mejoran la experiencia del usuario. Su integración es sencilla, su peso reducido y su uso es ampliamente documentado.

Backend

PHP (Backend)

PHP fue seleccionado como lenguaje de backend por su facilidad de aprendizaje, amplia documentación y compatibilidad con servidores gratuitos como InfinityFree. Además, PHP se integra fácilmente con MySQL y es muy adecuado para proyectos educativos y de prototipado rápido.

MySQL (XAMPP)

MySQL es un sistema de gestión de bases de datos relacional ampliamente utilizado. Se empleó junto con XAMPP para facilitar un entorno local completo y gratuito. MySQL permite estructurar los datos de forma eficiente y realizar consultas complejas con alto rendimiento.

Entorno de desarrollo

Visual Studio Code

VS Code se usó como entorno de desarrollo por su ligereza, extensibilidad y compatibilidad con múltiples tecnologías (Angular, PHP, HTML, CSS, etc.), la variedad de extensiones que se pueden aplicar agilizó el desarrollo.

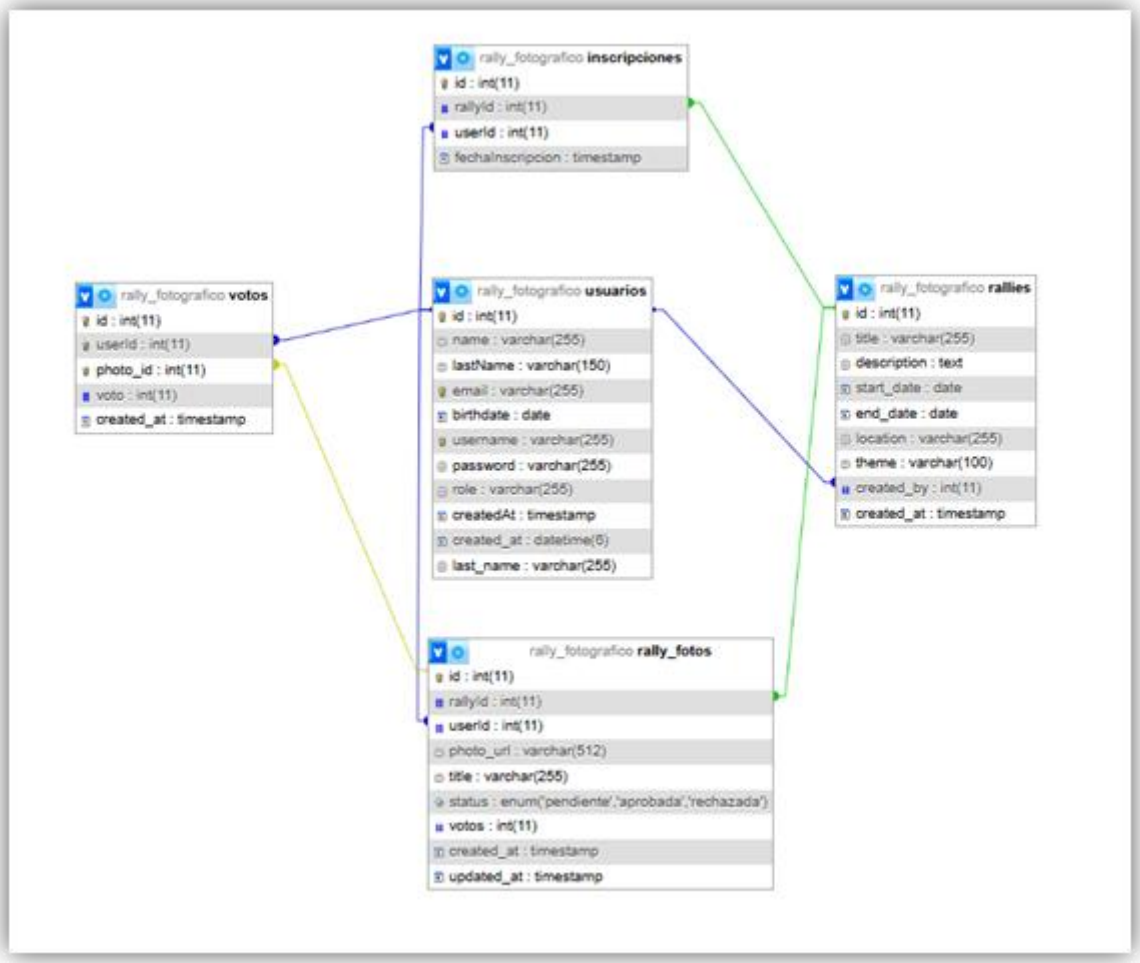
Despliegue

Despliegue en Wuaze(InfinityFree)

Se optó por InfinityFree por ofrecer un hosting gratuito con soporte para PHP y MySQL. Aunque limitado en algunos aspectos (como el tamaño de archivos o uso de .htaccess, no permite PUT y DELETE de forma directa), permite hacer un despliegue realista y accesible para una evaluación académica.

2. Diagrama entidad-relación (ER)

La base de datos de la aplicación web "Rally Fotográfico" está normalizada y organizada en torno a **seis entidades clave** que reflejan la estructura funcional del sistema: usuarios, rallies, inscripciones, fotografías, votos y relaciones administrativas.



Entidades y atributos principales

1. Usuarios

Representa a todos los usuarios registrados en la plataforma. Incluye tanto participantes como administradores.

- id (PK): Identificador único del usuario.
- name: Nombre del usuario
- lastName: Apellidos del usuario.
- email: Dirección de correo electrónico.
- username: Nombre de usuario único.
- password: Contraseña cifrada.
- birthdate: Fecha de nacimiento.
- role: Rol del usuario (participante, administrador).
- createdAt: Fecha de registro.

2. Rallies

Define cada rally fotográfico como evento separado.

- id (PK): Identificador del rally.
- title: Título del rally
- description: Descripción del evento.
- start_date: Fechas de inicio
- end_date: Fin del rally.
- location: Localización o ámbito geográfico.
- theme: Tema central del rally.
- created_by: (FK → usuarios.id): Admin creador del rally.
- created_at: Fecha de creación del evento.

3. Inscripciones

Tabla intermedia que gestiona la inscripción de usuarios en rallies específicos.

- id (PK): ID de la inscripción.
- rallyId: (FK → rallies.id): Rally al que se inscribe.
- userId: (FK → usuarios.id): Usuario inscrito.
- fechaInscripcion: Marca temporal de inscripción.

Relación muchos a muchos entre usuarios y rallies (usuario puede participar en varios rallies y un rally tiene varios participantes).

4. Rally_fotos

Contiene las fotografías subidas por los participantes.

- id (PK): ID de la foto.
- rallyId: (FK → rallies.id): Rally en el que se presenta la foto.
- userId: (FK → usuarios.id): Autor de la foto.
- photo_url: Ruta de almacenamiento de la imagen.
- title: Título de la fotografía.
- status: Estado de la foto (pendiente, aprobada, rechazada).
- votos: Número total de votos recibidos.
- created_at, updated_at: Marcas de tiempo para control de cambios.

5. Votos

Registra cada voto emitido por usuarios del público (con cuenta registrada).

- id (PK): ID del voto.
- userId: (FK → usuarios.id): Usuario que emite el voto.
- photo_id: (FK → rally_fotos.id): Fotografía votada.
- voto: Valor del voto (presumiblemente binario o numérico).
- created_at: Fecha del voto.

  Cada usuario puede votar múltiples fotos, pero la aplicación puede implementar lógica para limitar votos.

Relaciones clave del modelo:

- **Un usuario puede inscribirse en múltiples rallies**, y viceversa → relación inscripciones.
- **Un rally contiene muchas fotografías**, subidas por los usuarios inscritos → relación rally_fotos.
- **Cada foto puede recibir muchos votos**, y cada usuario puede emitir muchos votos → relación votos.
- **Un usuario puede ser administrador** y crear rallies (campo created_by en rallies).

Entidades principales:

- **Usuario**
  - id_usuario (PK)
  - nombre
  - apellidos
  - email
  - contraseña (hash)
  - rol (participante, administrador)
  - fecha_registro
- **Fotografía**
  - id_foto (PK)
  - id_usuario (FK)
  - titulo
  - descripcion
  - archivo (ruta)
  - fecha_subida
  - estado (pendiente, admitida, rechazada)
- **Votación**
  - id_voto (PK)
  - id_foto (FK)
  - id_usuario (FK) *(público votante)*
  - fecha_voto
- **Configuración del sistema:**
  No existe una tabla genérica de configuración. Todos los parámetros relevantes (fechas, límite de participantes, tema, ubicación, etc.) están definidos en la tabla rallies. Esto simplifica la gestión y evita duplicación innecesaria de datos.

Relaciones:

- Un **usuario** puede subir varias **fotografías**.
- Un **usuario** del tipo “público” puede votar muchas **fotografías**.
- Cada **fotografía** puede recibir muchos **votos**.
- Un **administrador** modifica parámetros desde **configuración**.

3. Diagrama de arquitectura de la aplicación

Aunque el backend PHP no implementa el patrón **MVC** formal, se ha estructurado de forma modular, agrupando funcionalidades en archivos separados como rallies.php, usuarios.php, fotos.php, etc.
Esto permite mantener el código organizado, aunque las funciones de controlador, modelo y salida están mezcladas dentro del mismo archivo.

La arquitectura está basada en el modelo **cliente-servidor** distribuido en tres capas:

1. Capa de presentación (Frontend):

- Desarrollada con Angular.
- Se encarga de la interfaz de usuario, validación básica de formularios, enrutamiento y llamadas HTTP.
- Comunicación asíncrona con el backend mediante **servicios Angular (HttpClient)**.

2. Capa de lógica de negocio (Backend):

- Implementada en PHP.
- Gestiona rutas API REST para:
    - Autenticación (login, registro).
    - Gestión CRUD de usuarios y fotografías.
    - Validación de fotos (admin).
    - Sistema de votación.
- Valida los datos recibidos del cliente antes de interactuar con la base de datos.

3. Capa de datos (Base de datos MySQL):

- Almacenamiento estructurado de usuarios, fotos y votos.
- Consultas SQL optimizadas.
- Control de integridad mediante claves foráneas.

Flujo general:

1. El usuario interactúa con la app Angular.
2. Angular envía peticiones HTTP al servidor PHP.
3. PHP valida los datos y realiza operaciones sobre MySQL.
4. PHP devuelve una respuesta (JSON) a Angular.
5. Angular actualiza la interfaz en función del resultado.

4. Diagrama de casos de uso (UML)

Actores:

- **Participante**
- **Administrador**
- **Público (logueado, sin fotos propias)**

Casos de uso por actor:

Participante:

- Registrarse
- Iniciar sesión
- Editar perfil
- Subir fotografía
- Consultar estado de sus fotografías
- Eliminar fotografía (si está pendiente o rechazada)
- Consultar ranking en la galería

Administrador:

- Iniciar sesión como admin
- Validar o rechazar fotografías
- Configurar reglas del rally (fechas, descripción, etc.)
- Gestionar usuarios
- Ver estadísticas y ranking general

Público general (logueado):

- Iniciar sesión
- Navegar por la galería
- Votar fotografías
- Ver resultados del ranking en la galería

5. Desarrollo del proyecto: decisiones técnicas, dificultades encontradas y soluciones

Decisiones técnicas clave:

- Se eligió PHP en el backend por su facilidad de despliegue en servidores gratuitos como InfinityFree y su rápida integración con MySQL, frente a opciones como Node.js que requieren configuraciones más complejas.
- Angular fue seleccionado para el frontend por su robustez y escalabilidad, frente a alternativas como Spring con java, priorizando la experiencia adquirida durante el curso.

Dificultades encontradas:

- **Comunicación Angular–PHP:**
  Angular usa peticiones HTTP basadas en JSON, mientras que PHP tradicionalmente trabaja con $_POST. Se resolvió utilizando cabeceras adecuadas (Content-Type: application/json) y decodificando el cuerpo con json_decode(file_get_contents('php://input')).
- **Validación de imágenes:**
  Fue necesario controlar tamaño, formato y seguridad. Se implementaron validaciones en ambos extremos (formulario Angular y PHP backend).
- **Problemas de CORS:**
  Al trabajar con frontend y backend en rutas diferentes, surgieron errores de política de origen cruzado. Se resolvió añadiendo las cabeceras Access-Control-Allow-Origin en PHP en un archivo único para evitar duplicar códigos.
- **Gestión del despliegue:**
  InfinityFree presentaba varias limitaciones técnicas, como la **imposibilidad de manejar archivos pesados**, la **restricción del uso de reglas avanzadas en .htaccess** y la **no aceptación de peticiones HTTP PUT o DELETE**.

  Para solventar esto último, se optó por una estrategia habitual en servidores compartidos: **enviar peticiones POST con un parámetro adicional (action)** indicando la operación deseada (por ejemplo, delete, update, etc.).

  En el backend PHP, se implementó un switch sobre el valor de $_POST['action'] para simular comportamientos equivalentes a PUT y DELETE, manteniendo así la lógica REST sin depender del método HTTP original.

Además, se adaptaron las rutas y se limitó el tamaño de las imágenes para garantizar compatibilidad con las restricciones del hosting gratuito.

Herramientas y recursos útiles:

- **GitHub** para control de versiones.
- **VS Code** con extensiones como Prettier, Angular Language Service, PHP Intelephense.
- **Postman** para probar rutas backend.
- **Bootstrap** (en pruebas de diseño).

6. Consideraciones de ampliación futura

1. **Cambio de localStorage a sessionStorage**
   Ahora mismo funciona con localStorage con lo cual hay que asegurase que se cierra la sesión antes de salir del navegador, lo correcto es que al cerrarlo también se cierre la sesión.

2. **Sistema de comentarios**
   Añadir la posibilidad de que el público comente o valore las fotos con texto (previa moderación) fomentaría la participación activa y retroalimentación creativa.

3. **Recuperación de contraseña**
   Actualmente si no recuerdas la contraseña debes volver a registrarte, con un nuevo usuario y una nueva contraseña, lo ideal es que tengas la opción de recuperar la contraseña mediante un correo electrónico.

4. **Autenticación social (Google, Facebook)**
   Permitir que los usuarios se registren o inicien sesión mediante sus cuentas sociales aceleraría el proceso de acceso y reduciría barreras de entrada, especialmente para votantes ocasionales.

5. **Panel de estadísticas avanzado para administradores**
   Incluir gráficos por categoría, evolución temporal de votos, y actividad de los usuarios mediante una librería como Chart.js o Google Charts.

6. **Subida masiva de imágenes / Drag & Drop**
   Actualmente, las fotos se suben de una en una. Se podría implementar una zona de arrastrar imágenes para subir varias al mismo tiempo, mejorando la experiencia del usuario.