TACTIX

Alejandro Pelaez, Pau Fort i Marc Jaen
2DAW - Desenvolupament d’Aplicacions Web

ÍNDICE

1. Idea del Proyecto

Este proyecto surge a partir de una necesidad real planteada por uno de nuestros profesores, que forma parte de un equipo de voleibol amateur. Nos propuso la idea de desarrollar una aplicación que ayudara a su equipo a organizarse mejor, ya que actualmente no cuentan con una herramienta específica que cubra todas sus necesidades de comunicación, coordinación de entrenamientos, partidos y alineaciones.

A partir de esta propuesta, decidimos enfocar nuestro proyecto final en el desarrollo de una aplicación web pensada tanto para equipos deportivos pequeños o amateur como para jugadores casuales que simplemente quieren jugar a fútbol. La aplicación incluye funciones básicas como creación de equipos, gestión de miembros, calendarios, chats generales, encuestas, estadísticas de partido, creación de partidos y torneos.

Este proyecto tiene un interés privado lucrativo, ya que además de resolver una necesidad concreta, incluye la posibilidad de incorporar funcionalidades premium, que podrían ser de pago para los equipos que deseen una experiencia más completa.

Aunque existen aplicaciones similares como Spondo, muchas están dirigidas a clubes grandes, requieren suscripción para acceder a funciones clave y no están adaptadas completamente a las necesidades de equipos más pequeños o informales. Además, algunas tienen limitaciones de idioma o de personalización.

El valor diferencial de esta aplicación está en su enfoque simple pero efectivo, pensado desde la experiencia real de un equipo amateur. Ofrece una herramienta accesible, funcional y adaptable, con la posibilidad de monetización a través de un modelo freemium: una versión gratuita con todas las funciones esenciales, y una versión premium con extras pensados para usuarios que deseen un nivel de gestión más profesional.

2. Análisis de Necesidades

---

1. **Gestión de usuarios y equipos**

    - Crear Usuario
        - Nombre, Email, Foto, Edad
    - Crear Equipo
        - Invitar a miembros, crear y gestionar equipos

    - Roles y permisos
        - **Administrador:** Acceso al backoffice con permisos para ver, editar y eliminar usuarios, equipos y eventos, así como bloquear usuarios. También crear torneos.
        - **Usuario**: Crear equipo, unirse a equipos, ver equipos, crear partidos y unirse a partidos. Al estar dentro de un equipo, el usuario obtiene un "subrol" o título, dependiendo de su función:
            - Jugador (incluye capitán): Ver información del equipo, ver anuncios, utilizar el chat (de equipo e individual), ver pizarra.
            - Capitán: Solo él puede invitar al equipo, crear encuestas, dibujar en la pizarra interactiva. Además podrá crear anuncios y participar en el chat de equipo.

2. **Comunicación y colaboración**

    - Chat general entre usuarios de un mismo equipo (todos los usuarios del equipo)
    - Chat individual de usuario a usuario (todos los usuarios)
    - Notificaciones
        - Para nuevos anuncios, mensajes, encuestas, recordatorios (todos los usuarios)
    - Compartir archivos en chat de equipo (todos los premium)

3. **Calendario y anuncios**

    - Postear anuncios (capitán, no premium)
        - Partidos, reuniones, etc
    - Calendario (todos con equipo)
    - Recordatorios automáticos (todos)
        - Notificaciones el día anterior y una hora antes del evento

4. **Encuestas y votaciones**

    - Crear encuestas (todos)
    - Votaciones rápidas (todos)

5. **Versión premium**

    - Pizarra interactiva
    - Poder unirse a torneos como equipo

6. **Mapa**

    - Muestra en un mapa la localización de torneos y eventos cercanos
    - Capacidad de filtraje por tipo de “evento”

3. Análisis Técnico

Para el desarrollo de esta aplicación se utilizarán las siguientes tecnologías:

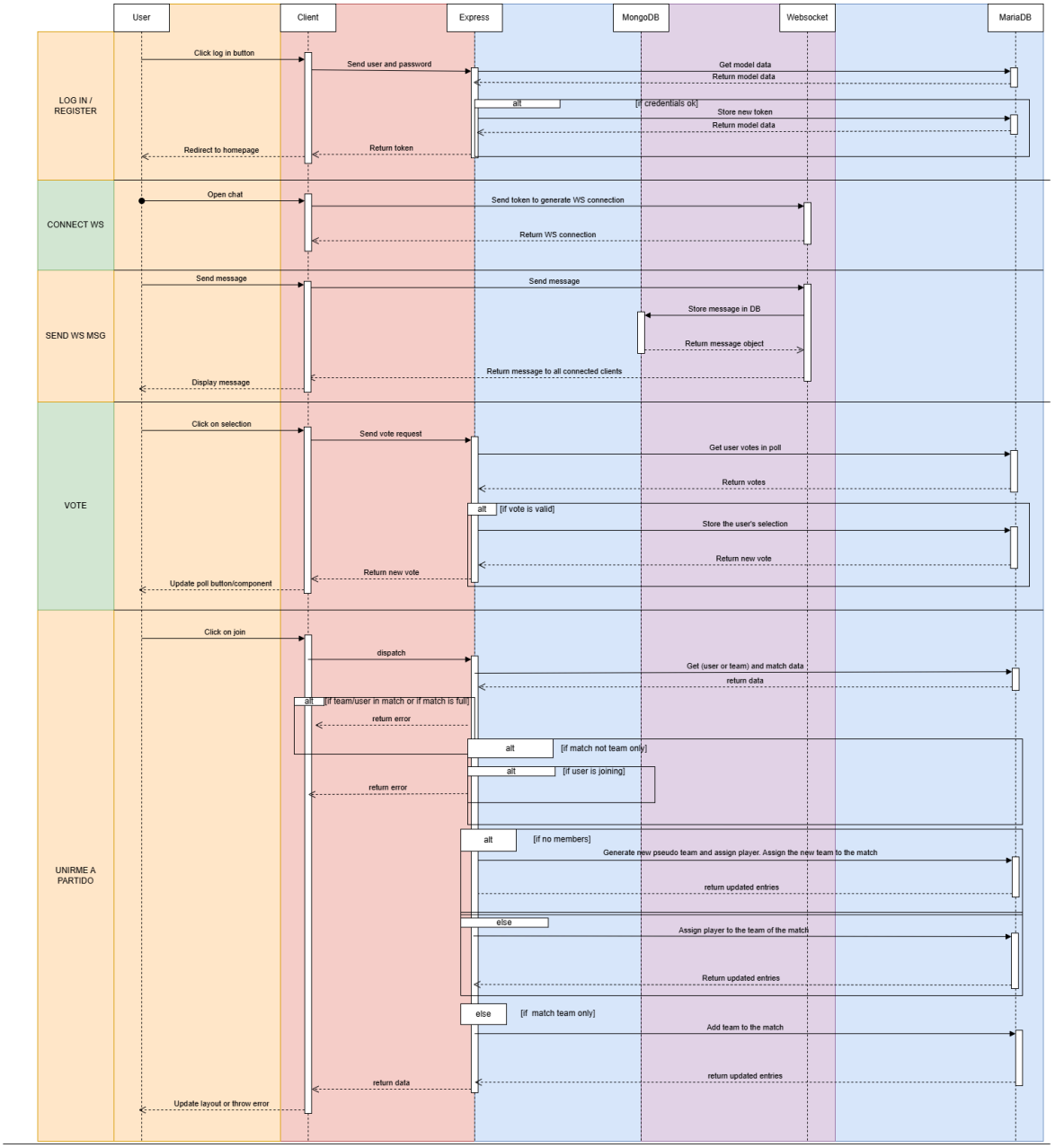
- **Lenguaje de programación:**
    - **Frontend:** JavaScript con React para la creación de una interfaz interactiva y dinámica. HTML5 para la estructura de la página web y Tailwind para los estilos de la misma.
    - **Backend:** Node.js con Express para gestionar las rutas y la lógica del servidor, aprovechando su capacidad para manejar peticiones en tiempo real y la integración con la base de datos
- **Tecnologías de servidor:**
    - **MongoDB** (NoSQL) por su flexibilidad en el manejo de datos no estructurados.
    - **Mongoose** para facilitar la interacción con la base de datos en Node.js.
    - **WebSocket** para la implementación de chats en tiempo real.
- **Autenticación y seguridad:**
    - **JWT (JSON Web Tokens)** para la autenticación de usuarios y roles.
    - **bcrypt** para la encriptación de contraseñas.

4. Propuesta del modelo de datos

**Propuesta del modelo de datos adjunto como archivo .pdf externo en la entrega*

5. Análisis y diseño de funcionalidades principales

*Análisis y diseño de funcionalidades principales adjunto como archivo .pdf externo en la entrega



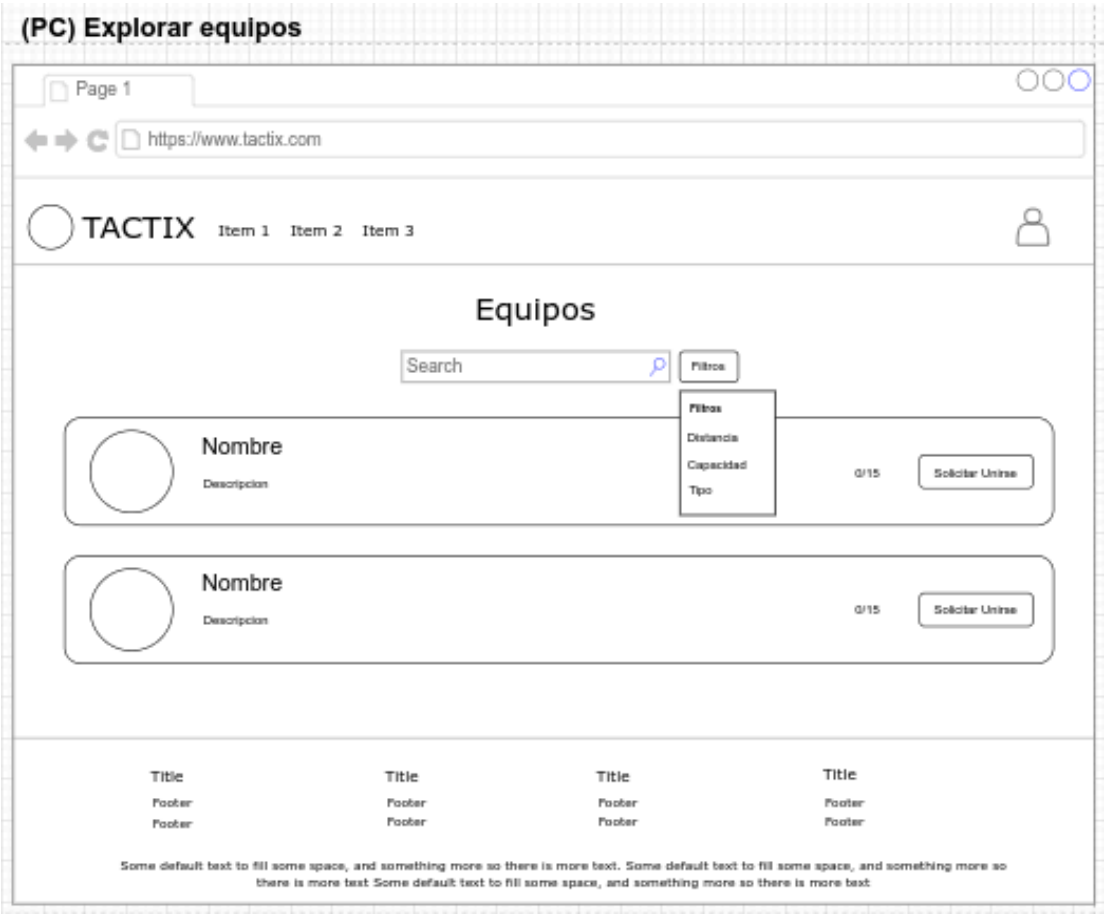
6. Diseño de funcionalidades principales

**Wireframes adicionales y Mapa de Navegación adjunto como archivos .pdf externos en la entrega*

Wireframes principales

(MB) Página principal con sesión

(PC) Explorar equipos

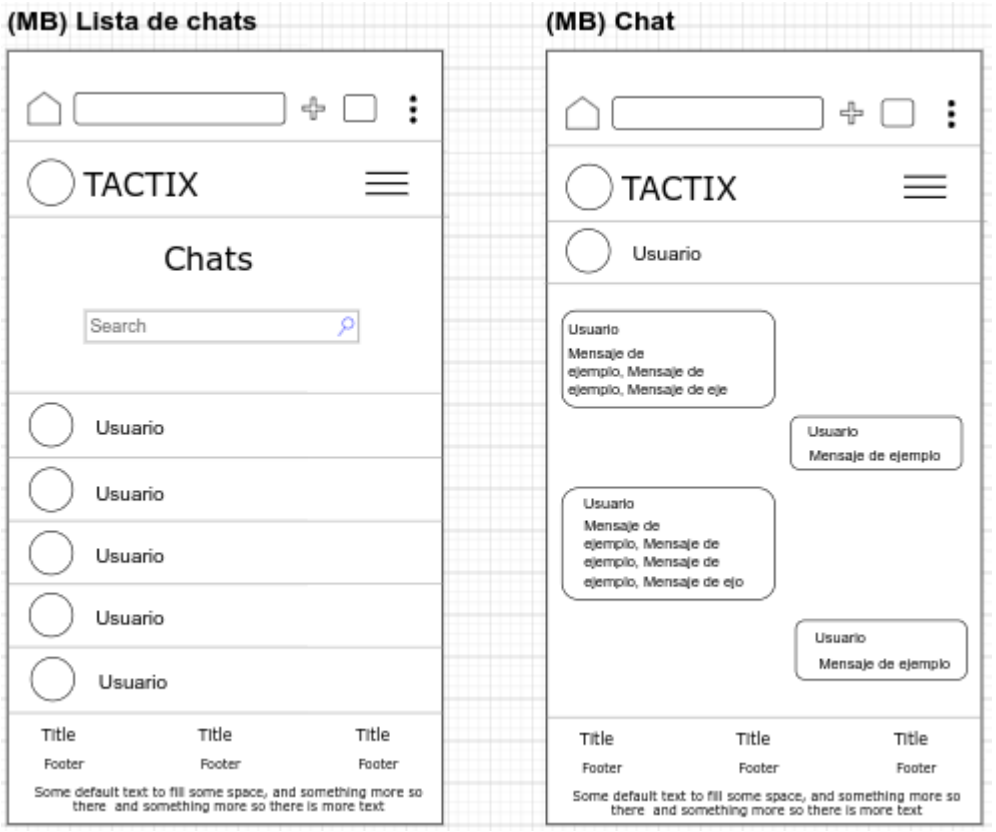


(MB) Explorar equipos

(PC) Página de encuestas

(MB) Página de encuestas

(PC) Chat



Guía de estilos

Tipografía

- Fuente principal: **Cal Sans**
- Fuente secundaria: Outfit
- Uso:
  - Títulos: Semi-bold o Bold, tamaño mayor (24px+)
  - Textos principales: Regular, tamaño normal (16px)
  - Pies de página, etiquetas y notas: Regular, tamaño pequeño (12px-14px)

Paleta de Colores (Coolors)

- Color Primario: #0057D9 (azul intenso)
- Color Secundario: #C1D8E1 (azul claro)
- Color de Contraste: #FF7F6B (salmón)
- Color de Fondo: #FFFFFF (blanco)
- Color de Texto: #161616 (negro suave

Estilos de Botones

- Primarios: Fondo color primario (#0057D9), texto blanco.
- Secundarios: Contorno color primario, texto color primario, fondo blanco.

- Contraste: Fondo color de contraste (#FF7F6B), texto blanco.

Iconografía

- Sistema de iconos: Material Icons o FontAwesome
- Estilo: Línea fina (Outline) o relleno, según el contexto.

Componentes Gráficos

- Inputs: Campos de formulario con esquinas redondeadas (border-radius: 8px), indicación de errores en color de contraste (#FF7F6B).
- Cards: Sombra suave, esquinas redondeadas (border-radius: 12px).
- Menús: Drop-downs sencillos con animación ligera.

Imágenes y Multimedia

- Estilo: Imágenes con bordes ligeramente redondeados, proporción respetada (object-fit: cover).
- Iconos en imágenes: Siempre integrados en SVG para mantener la calidad.

Responsividad

- Breakpoints principales:
    - Móvil: hasta 600px
    - Tablet: 601px – 960px
    - Escritorio: 961px+

Animaciones

- Animaciones suaves: Transiciones de 0,1s - 0,2s en hovers y cambios de estado.
- Tipos de animaciones: Principalmente “Fade in” y “Fade out”.

7. Implementación del modelo de datos

Modelo de Datos Relacional (SQL)

Tabla usuarios

- id INT NOT NULL AUTO_INCREMENT PRIMARY KEY
- nombre VARCHAR(100) NOT NULL
- email VARCHAR(30) NOT NULL UNIQUE
- contraseña VARCHAR(30) NOT NULL
- foto VARCHAR(255)
- fecha_nacimiento TIMESTAMP NOT NULL
- rol VARCHAR(10) NOT NULL (‘admin’ | ‘user’)

Tabla equipos

- id INT NOT NULL AUTO_INCREMENT PRIMARY KEY
- nombre VARCHAR(100) NOT NULL
- limite_miembros INT NOT NULL

Tabla miembros_equipo (relación n-m usuarios ↔ equipos)

- equipo_id INT NOT NULL, REFERENCES equipos(id) ON DELETE CASCADE
- usuario_id INT NOT NULL, REFERENCES usuarios(id) ON DELETE CASCADE
- titulo VARCHAR(20) NOT NULL (‘jugador’ | ‘capitán’)
- fecha_entrada TIMESTAMP NOT NULL
- PRIMARY KEY (equipo_id, usuario_id)

Tabla anuncios

- id INT NOT NULL AUTO_INCREMENT PRIMARY KEY
- equipo_id INT NOT NULL, REFERENCES equipos(id) ON DELETE CASCADE
- fecha TIMESTAMP NOT NULL
- contenido TEXT NOT NULL

Tabla encuestas

- id INT NOT NULL AUTO_INCREMENT PRIMARY KEY
- pregunta TEXT NOT NULL

Tabla opciones_encuesta

- encuesta_id INT NOT NULL, REFERENCES encuestas(id) ON DELETE CASCADE
- opcion VARCHAR(100) NOT NULL

- PRIMARY KEY (encuesta_id, opcion)

Tabla respuestas_encuesta

- id INT NOT NULL AUTO_INCREMENT PRIMARY KEY
- encuesta_id INT NOT NULL, REFERENCES encuestas(id) ON DELETE CASCADE
- usuario_id INT NOT NULL, REFERENCES usuarios(id) ON DELETE CASCADE
- respuesta VARCHAR(100) NOT NULL

Tabla torneos

- id INT NOT NULL AUTO_INCREMENT PRIMARY KEY
- fecha TIMESTAMP NOT NULL
- ubicacion_x INT NOT NULL
- ubicacion_y INT NOT NULL

Tabla torneo_equipos (relación n-m torneos ↔ equipos)

- torneo_id INT NOT NULL, REFERENCES torneos(id) ON DELETE CASCADE
- equipo_id INT NOT NULL, REFERENCES equipos(id) ON DELETE CASCADE
- PRIMARY KEY (torneo_id, equipo_id)

Tabla partidos

- id INT NOT NULL AUTO_INCREMENT PRIMARY KEY
- fecha_inicio TIMESTAMP NOT NULL
- ubicacion_x INT NOT NULL
- ubicacion_y INT NOT NULL
- mvp_usuario_id INT, REFERENCES usuarios(id)

Columnas adicionales en partidos para los dos equipos:

- equipo1_id INT, REFERENCES equipos(id)
- equipo2_id INT, REFERENCES equipos(id)

Tabla goles

- id INT NOT NULL AUTO_INCREMENT PRIMARY KEY
- partido_id INT NOT NULL, REFERENCES partidos(id) ON DELETE CASCADE
- usuario_id INT NOT NULL, REFERENCES usuarios(id) ON DELETE CASCADE
- minuto INT NOT NULL

Tabla invitaciones_equipo

- id INT NOT NULL AUTO_INCREMENT PRIMARY KEY
- equipo_id INT NOT NULL REFERENCES equipos(id) ON DELETE CASCADE
- invitador_id INT NOT NULL REFERENCES usuarios(id) ON DELETE RESTRICT
- invitado_id INT NOT NULL REFERENCES usuarios(id) ON DELETE CASCADE
- estado VARCHAR(10) NOT NULL CHECK (estado IN ('pendiente','aceptada','rechazada'))
- fecha_invitacion TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
- fecha_respuesta TIMESTAMP

Invitaciones

- id INT NOT NULL AUTO_INCREMENT PRIMARY KEY
- equipo_id INT NOT NULL REFERENCES equipos(id) ON DELETE CASCADE
- invitador_id INT NOT NULL REFERENCES usuarios(id) ON DELETE RESTRICT
- invitado_id INT NOT NULL REFERENCES usuarios(id) ON DELETE CASCADE
- estado VARCHAR(10) NOT NULL DEFAULT 'pendiente' CHECK (estado IN ('pendiente','aceptada','rechazada'))
- creado_en TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
- respondido_en TIMESTAMP

Modelo de datos NO relacional (MongoDB)

Chat

- tipo String, enum [“individual”, “grupo”], required
- usuario1_id Int (referencia a usuarios.id en SQL) — requerido si tipo=individual
- usuario2_id Int (idem)
- equipo_id Int (referencia a equipos.id en SQL) — requerido si tipo=grupo
- mensajes (array con objetos mensaje)
    - autor String (usuarios.id de SQL), required
    - contenido String, required
    - enviado_en Date, default Date.now
- creado_en Date, default Date.now

Notificacion

- usuario_id Int (usuarios.id de SQL), required
- titulo String, required
- contenido String, required
- fecha Date, default Date.now
- leida Boolean, default false

8. Planificación de tareas y estimación de su duración

*Tablero de Trello adjunto en el siguiente enlace: *Trello*

9. Desarrollo y desplegamiento

*Enlace al repositorio de nuestro trabajo: *GitHub*
*Enlace del proyecto desplegado: *TACTIX*

10. Ejecución del proyecto

Al principio fuimos muy ambiciosos al planificar numerosas funcionalidades, pero no pudimos implementarlas todas, principalmente debido a la reducción del tiempo disponible.

En general, nos adaptamos bien a los plazos que habíamos establecido. Durante las dos primeras semanas cumplimos con todas las tareas programadas, finalizándolas dentro del tiempo acordado. Sin embargo, al reducir el tiempo total del proyecto de cuatro a tres semanas, nos vimos obligados a reorganizar el trabajo. Aunque completamos correctamente las tareas de la tercera semana y tratamos de incorporar también las previstas para la cuarta, no fue posible abarcarlo todo.

La mayoría de las funcionalidades se desarrollaron correctamente, aunque algunas, como la de torneos, quedaron incompletas.

Inicialmente queríamos que los equipos pudieran inscribirse en torneos y competir entre ellos. Conseguimos implementar la creación de torneos por parte de los administradores y la posibilidad de que los equipos se unieran. También se desarrolló la visualización de los brackets (cuadro de competición), pero no llegamos a implementar el avance de fases ni la lógica completa del torneo.

Otra funcionalidad que nos hubiera gustado añadir es una versión premium, en la que se restringieran algunas características para los usuarios suscritos. Esto nos habría dado la posibilidad de monetizar el proyecto en algún momento, ofreciendo funcionalidades avanzadas a cambio de una cuota mensual.

Durante el desarrollo hemos tenido algunas complicaciones técnicas. Por ejemplo, inicialmente no era posible que un usuario se uniera a un partido de forma individual, ya que era obligatorio referenciar a un equipo.
Para solucionarlo decidimos que al unirse a un partido, el sistema genera un pseudoequipo. Para ello, añadimos una columna *pseudoequipo* en la base de datos de equipos, cuyo valor podía ser *true* o *false*. Si era *true*, el equipo no aparecía en el listado general de equipos.

La parte de equipos supuso varias complicaciones en la recta final. El usuario podía tener valor “null” en la columna de id de equipo pero estar en varios equipos a la vez, cuando en teoría el jugador solamente podía estar en un único equipo. Esto daba problemas a la hora de visualizar información tanto de usuarios como de los mismos equipos, así como trabas en el funcionamiento de la aplicación. Para solucionarlo, lo que hicimos fue modificar los filtros de las consultas a la base de datos, ya que los filtros actuales solo revisaban que la membresía del equipo estuviese activa y no que fuese de un pseudoequipo o un “equipo escondido” (equipo que se ha quedado sin jugadores en el cual el jugador actual había estado anteriormente). En resumen, la incompetencia de los filtros a la base de datos provocaba que se mostrase información de equipos de los que el usuario había formado parte en algún momento.

Con el objetivo de mejorar la apariencia de la aplicación decidimos utilizar iconos en formato ".svg" para representar visualmente ciertas características. No obstante, la composición de este tipo de archivos parecen tener cierta incompatibilidad con el framework que estábamos usando en el frontend; React. Para solucionar el problema buscamos herramientas por internet para adaptar los svg a la sintaxis de react. La página que más nos ayudó fue "Svg2jsx", aunque por el camino encontramos otras páginas como "SvgViewer", pero que no eran 100% eficaces. Posteriormente creamos un nuevo componente en el que definíamos y exportábamos los svg como variables con el fin de importarlos con más facilidad en el resto de los archivos.

Trabajando en el backoffice nos dimos cuenta de que no había apenas notificaciones que indicasen cómo iban los procesos del backend. Para ello nos pusimos a construir un método de aparición de mensajes flash de éxito y error. En primer lugar lo intentamos hacer modificando la url de redirección mediante "querystring" al realizar la consulta sql y parecía funcionar correctamente. No obstante, al recargar la página, el mensaje flash persistía (ya que estaba recargando la url modificada todo el rato), cosa que no quería que pasase. Tuve que recurrir a una librería de Javascript llamada "express-session", la cual almacena datos en una sesión y luego los borra. De esta forma, la aparición y desaparición de mensajes flash pasó a funcionar correctamente.

El día 13 de mayo empezamos a trabajar en el mapa en el que los usuarios podrían ver la localización de los partidos y torneos existentes. Para conseguir la dirección correspondiente a las coordenadas de cada evento necesitábamos una api externa, así que nos pusimos a investigar. En primer lugar encontramos una la cual sólo aceptaba una petición cada mucho tiempo, por lo que no era viable hacer uso de ella. Después de un rato dimos con "LocationIQ", api que se adaptaba mejor a nuestras necesidades, pues la versión gratuita de esta admite 5.000 peticiones al día. No obstante, al empezar a probar su funcionamiento nos encontramos con que estábamos obteniendo un error que indicaba una limitación muy reducida en el número de peticiones que podía realizar la aplicación al momento. Por esta razón tuvimos que buscar un método para solucionar el problema, y así fue como descubrimos un paquete de Javascript llamado "Bottleneck", cuya utilidad es controlar la cantidad de peticiones "http" que se realizan en cierto periodo de tiempo. Gracias a esta librería ahora podemos realizar más llamadas a la api de forma simultánea, de manera que la aplicación es usable para el público sin problemas de por medio.

También tuvimos problemas con la autentificación. Usábamos una misma cookie para guardar el token en el frontend y el backend, por lo que al trabajar en localhost, permitía acceder al backoffice sin ser administrador, simplemente con iniciar sesión desde el frontend podías acceder.
Para solucionarlo, implementamos una validación adicional en el backoffice, comprobando en todas la rutas que el usuario tuviera el rol de administrador

Además, experimentamos un problema de cruce de cuentas entre el frontend y el backend. Si se mantenían sesiones abiertas con diferentes usuarios en ambas interfaces, las peticiones realizadas desde el frontend se ejecutaban bajo el usuario del backoffice, no el usuario real.

Lo resolvimos ajustando la lógica del middleware de autenticación: anteriormente se daba prioridad a las cookies del backend y, si existían, no se validaba el token enviado en la petición. Ahora, si una petición incluye un token, este se valida con prioridad sobre cualquier cookie del backend.

Durante el proceso de desarrollo, nos dimos cuenta de que la paleta de colores que habíamos elegido representaba bien la idea del proyecto, así que decidimos rehacerla y nos decantamos por esta combinación de colores: https://coolors.co/1f2937-f3f4f6-eab308.

11. Conclusiones

El desarrollo de TACTIX ha sido una experiencia enriquecedora tanto a nivel técnico como organizativo. Aunque no hayamos seguido la idea original porque era algo mucho más ambicioso, hemos conseguido derivarla a un sector mucho más casual y accesible para nosotros. A partir de una necesidad real, identificamos una oportunidad para diseñar una solución práctica, funcional y escalable, orientada a equipos deportivos amateur y jugadores ocasionales que buscan una herramienta eficaz de organización y comunicación.

Durante el proceso, logramos implementar con éxito la mayoría de las funcionalidades clave: creación y gestión de equipos, chats individuales y grupales, calendarios de eventos, encuestas, anuncios, y la integración de una base de datos híbrida (relacional y no relacional) para gestionar tanto la estructura de datos como la comunicación en tiempo real.

Sin embargo, también enfrentamos limitaciones de tiempo que impidieron completar ciertas funcionalidades, como la lógica completa de los torneos o el sistema de suscripciones premium. A pesar de ello, dejamos sentadas las bases técnicas y conceptuales para su desarrollo futuro, lo que garantiza la escalabilidad del proyecto.

En definitiva, TACTIX representa una solución sólida, accesible y adaptable a las necesidades reales de los usuarios. Su arquitectura y diseño permiten su evolución hacia una plataforma más completa y profesional, con opciones de monetización que podrían garantizar su sostenibilidad a largo plazo. Este proyecto no solo ha sido un ejercicio técnico, sino también una muestra de nuestra capacidad de colaboración, análisis y adaptación ante los imprevistos, sentando un precedente positivo para futuros desarrollos.

Una de las mejoras que podríamos implementar en el futuro sería completar la funcionalidad de los torneos, ya que, por falta de tiempo, no llegaron a ser plenamente operativos. Actualmente, los equipos pueden unirse, pero no es posible avanzar en las fases del torneo.

Nos habría gustado incorporar diferentes tipos de torneos, tanto gratuitos como de pago, con la posibilidad de ofrecer premios para los ganadores. Esta funcionalidad no sólo ampliaría las opciones de participación, sino que también podría aumentar el interés y la competitividad entre los usuarios. Otra propuesta de mejora sería el desarrollo de una versión premium de la aplicación, en la que ciertas funcionalidades estuvieran reservadas para los usuarios suscritos mediante el pago de una cuota mensual. Esto permitiría monetizar el proyecto y ofrecer servicios adicionales de valor.

Finalmente, una de las ideas que tuvimos al principio fue la de abarcar todo tipo de deportes con nuestra aplicación. Pero, de nuevo, por falta de tiempo hemos tenido que reducir la carga de trabajo y orientarlo solamente al fútbol, puesto que pensamos que es el deporte más popular en nuestro país. Nos hubiera gustado abarcar bastante más, pero estamos contentos con el resultado final.

12. Webgrafía

Svg2jsx. https://svg2jsx.com/

SvgViewer. https://www.svgviewer.dev/

Tailwind Labs. Tailwind Css (2019). https://tailwindcss.com/

OpenAI. Chat GPT (2022). https://chat.openai.com/

HighFlyer. Deepseek (2025). https://chat.deepseek.com/

Tj Holowaychuk. ExpressJS. Guide (2010). https://expressjs.com/en/guide/routing.html

Meta. React. Learn React (2013). https://react.dev/learn

NPM. Bottleneck package (2019). https://www.npmjs.com/package/bottleneck

Anexo 1

Requisitos del sistema

Requisitos de software:

- Sistema Operativo
  - Windows 10/11
  - macOS 12 o superior
  - Linux (Ubuntu 20.04 o superior, Debian, Fedora, etc.)

- Requisitos generales
  - **Node.js**: v18.x o superior
  - **npm**: v9.x o superior
  - **Base de datos**
    - MongoDB
    - MariaDB

- Frontend (React)
  - **Framework**: React 19.x
  - **Herramientas**
    - TailwindCSS
    - React Router DOM
    - Leaflet

- Backend (Node.js + Express)
  - **Framework**: Express 5.x
  - **Autenticación**: JWT, bcrypt
  - **Bases de datos**
    - MongoDB (v6+)
    - MariaDB (v10.5+)
  - **Middlewares**
    - cors
    - cookie-parser
    - express-session
    - method-override
    - multer

Requisitos de Hardware:

- Recomendados
  - **CPU**: Intel i5 de 6ta gen o AMD equivalente
  - **RAM**: 8 GB
  - **Almacenamiento**: 2 GB

Instalación

El proyecto está dividido en dos partes: frontend y backend. Cada una tiene su propia carpeta y sus propias dependencias que deben instalarse por separado. Una vez instaladas las dependencias, se deben iniciar ambas partes del proyecto.

1. Clonar el repositorio:
    a. git clone git@github.com:jmirinformatica/projecte-final-equip08.git
2. Entrar en la carpeta del frontend y instalar las dependencias:
    a. npm install --force
3. Entrar en la carpeta del backend y instalar las dependencias:
    a. npm install
4. Crear los archivos .env en el frontend y en el backend:
    a. Copia esto en tu archivo .env del frontend:

```
PORT = 8000
```

    b. Copia esto en tu archivo .env del backend:

```
PORT=3000
MONGODB_USER=mjaeng
MONGODB_PSWD=Ly7sOkHiIbMibdQi1d17pzu
MONGODB_HOST=cluster0.l3t8s.mongodb.net
MONGODB_NAME=""
MONGODB_URI="mongodb+srv://mjaeng:Ly7sOkHiIbMibdQi1d17pzu@cluster0.l3t8s
.mongodb.net/"

DB_HOST=arrakis.insjoaquimmir.cat
DB_PORT=13306
DB_USER=equip08
DB_PSWD=WB4mdRM8Pc
DB_NAME=equip08_db2

JWT_SECRET=1234

LOCATIONIQ_KEY=pk.6c8b3bcc22fff5da29ea9f54c2bac4d1
```

5. Ejecutar el proyecto:
    a. En las carpetas de frontend y backend:
        i. npm run dev
6. Accede a las rutas
    a. accede a http://localhost:8000 para entrar al cliente
    b. accede a http://localhost:3000 para entrar al backoffice