[DOT & DOT] Diseño del Frontend

Este documento describe la arquitectura y estructura del frontend de la aplicación **DOT & DOT**, desarrollada con React 19 y Vite.

Framework: React 19
Build Tool: Vite
Routing: React Router DOM v7
Estado: React Hooks (useState, useEffect, useContext)
Estilos: CSS Modules
Comunicación: Fetch API, WebSocket (STOMP)

1. Estructura del Proyecto

- **dotanddot-web/**
  - **public/** - Recursos estáticos públicos
    - `ads/` - Recursos para publicidad
    - `login/` - Recursos de login
    - `welcome/` - Recursos de bienvenida
  - **src/** - Código fuente de la aplicación
    - **assets/** - Imágenes y recursos estáticos
    - **components/** - Componentes reutilizables
      - `Logo/` - Componente del logotipo
      - `LogoBall/` - Animación de pelota
    - **css/** - Estilos globales
      - `colors.css` - Sistema de colores
      - `index.css` - Estilos base
    - **js/** - Lógica de negocio y servicios
      - `auth.mjs` - Servicio de autenticación
      - `env.js` - Variables de entorno y endpoints
      - `http.js` - Cliente HTTP
      - `session.mjs` - Gestión de sesión
      - `cruds/` - Servicios CRUD por entidad
      - `home/` - Servicios específicos del home
    - **pages/** - Páginas principales de la aplicación
      - `Admin/` - Panel de administración
      - `App/` - Aplicación principal
      - `Error/` - Página de error
      - `Login/` - Autenticación
      - `PrivacyPolicy/` - Política de privacidad
      - `Referee/` - Panel de árbitro
      - `UserFavorites/` - Favoritos del usuario
    - `AuthGuard.jsx` - Guard de autenticación general
    - `AuthRefereeGuard.jsx` - Guard para árbitros
    - `Index.jsx` - Router principal
    - `main.jsx` - Punto de entrada de la aplicación
  - `index.html` - HTML base
  - `package.json` - Dependencias y scripts
  - `vite.config.js` - Configuración de Vite

2. Arquitectura del Frontend

2.1 Patrón de Arquitectura

La aplicación sigue una **arquitectura basada en componentes** con separación de responsabilidades:

- **Páginas** (`pages/`): Componentes de nivel superior que representan rutas
- **Componentes** (`components/`): Componentes reutilizables
- **Servicios** (`js/`): Lógica de negocio y comunicación con backend
- **Guards**: Protección de rutas basada en autenticación y roles

2.2 Capa de Servicios

Servicio HTTP (`http.js`)

```
- get(url, token)           // GET request
- post(url, body, token)    // POST request
- put(url, body, token)     // PUT request
- del(url, token)           // DELETE request
```

Autenticación (`auth.mjs`)

```
- login(username, password)
- register(userData)
- logout()
- validateSession()
- validateRole(role)
- forgotPassword(email)
- resetPassword(token, newPassword)
```

Sesión (`session.mjs`)

```
- saveSession(data)
- getSession()
- clearSession()
- isAuthenticated()
```

Servicios CRUD (`js/cruds/`)

- `cities.mjs` - Gestión de ciudades
- `clubs.mjs` - Gestión de clubes
- `coaches.mjs` - Gestión de entrenadores
- `competition.mjs` - Gestión de competiciones
- `gameplaces.mjs` - Gestión de pabellones
- `leagues.mjs` - Gestión de ligas
- `persons.mjs` - Gestión de personas
- `players.mjs` - Gestión de jugadores
- `referees.mjs` - Gestión de árbitros
- `roles.mjs` - Gestión de roles
- `seasons.mjs` - Gestión de temporadas
- `teams.mjs` - Gestión de equipos
- `users.mjs` - Gestión de usuarios

3. Sistema de Rutas

3.1 Rutas Públicas

```
/                           // Página principal
/login                      // Autenticación
/register                   // Registro de usuarios
/verify-account/:token      // Verificación de email
/forgot-password            // Recuperación de contraseña
/reset-password/:token      // Reseteo de contraseña
/privacy-policy             // Política de privacidad
/app                        // Dashboard principal
/app/home                   // Inicio
/app/clubs                  // Listado de clubes
/app/competitions           // Competiciones
/app/competition/:id        // Detalle de competición
```

3.2 Rutas Protegidas (Usuario Autenticado)

```
/app/contact                // Contacto
/app/profile                // Perfil de usuario
/app/favorites              // Favoritos del usuario
```

3.3 Rutas de Administrador (ROLE_ADMIN)

```
/admin                      // Panel de administración
/admin/cities               // Gestión de ciudades
/admin/clubs                // Gestión de clubes
/admin/coaches              // Gestión de entrenadores
/admin/competitions         // Gestión de competiciones
/admin/games                // Gestión de partidos
/admin/gameplaces           // Gestión de pabellones
/admin/leagues              // Gestión de ligas
/admin/persons              // Gestión de personas
/admin/players              // Gestión de jugadores
/admin/referees             // Gestión de árbitros
/admin/roles                // Gestión de roles
/admin/seasons              // Gestión de temporadas
/admin/teams                // Gestión de equipos
/admin/users                // Gestión de usuarios
```

3.4 Rutas de Árbitro (ROLE_REFEREE)

```
/referee                    // Panel de árbitro
/referee/games              // Listado de partidos asignados
/referee/game/:code         // Arbitrar partido en vivo
```

4. Componentes Principales

4.1 Guards de Autenticación

AuthGuard

Protege rutas que requieren autenticación:

- Verifica token de sesión
- Redirige a `/login` si no está autenticado
- Valida sesión con el backend

AuthRefereeGuard

Protege rutas específicas de árbitros:

- Verifica autenticación
- Valida rol `ROLE_REFEREE`
- Redirige a `/app` si no tiene permisos

AuthAdminGuard

Protege rutas de administración:

- Verifica autenticación
- Valida rol `ROLE_ADMIN`
- Redirige según permisos

4.2 Componentes Reutilizables

Logo

Componente del logotipo de la aplicación con estilos personalizados.

LogoBall

Componente del logotipo de la aplicación que incluye la serigrafía oficial de la aplicación con una redirección al home.

5. Módulos Principales

5.1 Módulo de Login

Componentes:

- `Login.jsx` - Contenedor principal
- `LoginForm.jsx` - Formulario de inicio de sesión
- `RegisterForm.jsx` - Formulario de registro
- `ValidateAccount.jsx` - Verificación de email

Funcionalidades:

- Login con username/password
- Registro de nuevos usuarios
- Verificación de email con token
- Recuperación de contraseña
- Validación de formularios

5.2 Módulo de Administración

Estructura:

- **Admin/** - Módulo de administración
  - `Admin.jsx` - Contenedor principal
  - `Admin.css` - Estilos del panel
  - **components/** - Componentes del panel
    - **buttons/** - Botones reutilizables
      - `cancel/` - Botón de cancelar
      - `create/` - Botón de crear
      - `delete/` - Botón de eliminar
      - `edit/` - Botón de editar
      - `new/` - Botón de nuevo
      - `update/` - Botón de actualizar
  - **partials/** - Vistas CRUD por entidad
    - `City/` - Gestión de ciudades
    - `Clubs/` - Gestión de clubes
    - `Coach/` - Gestión de entrenadores
    - `Competition/` - Gestión de competiciones
    - `Game/` - Gestión de partidos
    - `Gameplace/` - Gestión de pabellones
    - `League/` - Gestión de ligas
    - `Person/` - Gestión de personas
    - `Player/` - Gestión de jugadores
    - `Referee/` - Gestión de árbitros
    - `Roles/` - Gestión de roles
    - `Seasons/` - Gestión de temporadas
    - `Team/` - Gestión de equipos
    - `Users/` - Gestión de usuarios
    - `WelcomeAdmin/` - Pantalla de bienvenida
  - **screen-resolution-error/** - Error de resolución

Funcionalidades por entidad:

- **Listado**: Tabla con todas las entidades
- **Crear**: Formulario de creación
- **Editar**: Formulario de edición
- **Eliminar**: Confirmación y eliminación
- **Búsqueda**: Filtrado en tiempo real
- **Paginación**: Navegación por páginas

5.3 Módulo de Aplicación Principal

Estructura:

- **App/** - Módulo de aplicación principal
  - `App.jsx` - Contenedor principal con navegación
  - `App.css` - Estilos del layout principal
  - **components/** - Componentes de la aplicación
    - **clubs/** - Vista de clubes
      - `Clubs.jsx` - Listado de clubes con mapa interactivo
      - `Clubs.css` - Estilos para cards y layout
      - **partials/** - Subcomponentes
        - `ExtremaduraMap.jsx` - Mapa interactivo con geolocalización de clubes
        - `ExtremaduraMap.css` - Estilos del mapa (markers, zoom, tooltips)
        - **ClubView/** - Vista detallada de club
          - `ClubView.jsx` - Información completa del club con mapa
          - `ClubView.css` - Estilos para información de contacto y mapa
    - **competitions/** - Gestión de competiciones
      - `Competitions.jsx` - Lista de competiciones activas y finalizadas
      - `Competitions.css` - Estilos de tarjetas de competiciones
    - **contact/** - Formulario de contacto
      - `Contact.jsx` - Formulario con validación
      - `Contact.css` - Estilos del formulario
    - **home/** - Página de inicio
      - `Home.jsx` - Dashboard con partidos destacados
      - `Home.css` - Estilos de la página principal
      - **partials/** - Componentes del home
        - `Matches.jsx` - Listado de todos los partidos
        - `NextMatches.jsx` - Próximos partidos (tabla)
        - `RecentMatches.jsx` - Partidos recientes (tabla)
    - **profile/** - Perfil de usuario
      - `Profile.jsx` - Edición de datos personales
      - `Profile.css` - Estilos del formulario de perfil
    - **showCompetition/** - Detalle de competición
      - `ShowCompetition.jsx` - Vista completa con clasificación, filtros y partidos
      - `ShowCompetition.css` - Estilos para tabla de clasificación y filtros

Funcionalidades:

- **Visualización**: Competiciones activas con filtros por estado, equipo y fechas
- **Clubes**: Mapa interactivo de Extremadura con ubicación de clubes, zoom y detalles
- **Clasificación**: Tabla de posiciones con estadísticas (PJ, PG, PP, SG, SP, Dif, Pts)
- **Favoritos**: Sistema completo con botón de favoritos en cada partido, sincronización con backend
- **Búsqueda**: Filtrado en tiempo real por múltiples criterios
- **Responsive**: Adaptación completa a dispositivos móviles y tablets

5.4 Módulo de Árbitro

Componentes:

- `Referee.jsx` - Panel principal
- `RefereeGames.jsx` - Lista de partidos
- `LiveRefereeGame.jsx` - Arbitraje en vivo

Funcionalidades:

- Listado de partidos asignados
- Iniciar partido
- Anotar puntos por set
- Cambiar alineaciones
- Registrar sanciones
- Finalizar partido
- Sincronización en tiempo real (WebSocket)

API de Árbitro (`refereeGameApi.mjs`):

```
- fetchGameByCode(uniqueCode)
- startGame(uniqueCode)
- finishGame(uniqueCode)
- updateSetPoints(uniqueCode, setNumber, pointsLocal, pointsVisit)
- addGameSanction(uniqueCode, sanctionData)
- getSetsArray(game)
```

6. Comunicación con el Backend

6.1 Endpoints API

Los endpoints están centralizados en `env.js`:

```
const BACK_IP = import.meta.env.VITE_BACK_IP || 'http://localhost:8080';

API = {
  AUTH: {
    LOGIN: `${BACK_IP}/auth/login`,
    REGISTER: `${BACK_IP}/auth/register`,
    VERIFY: (token) => `${BACK_IP}/auth/verify/${token}`,
    LOGOUT: `${BACK_IP}/auth/logout`,
    VALIDATE_SESSION: `${BACK_IP}/auth/validate-session`,
    VALIDATE_ROLE: (role) => `${BACK_IP}/auth/validate-role/${role}`,
    FORGOT_PASSWORD: `${BACK_IP}/auth/forgot-password`,
    RESET_PASSWORD: `${BACK_IP}/auth/reset-password`
  },
  ADMIN: {
    // Endpoints CRUD para todas las entidades
  },
  REFEREE: {
    GAMES: `${BACK_IP}/referee/matches`,
    GET_GAME: (uniqueCode) => `${BACK_IP}/referee/matches/${uniqueCode}`,
    START_GAME: (uniqueCode) => `${BACK_IP}/referee/start/${uniqueCode}`,
    FINISH_GAME: (uniqueCode) => `${BACK_IP}/referee/finish/${uniqueCode}`,
    UPDATE_POINTS: (uniqueCode, setNumber) =>
      `${BACK_IP}/referee/update-points/${uniqueCode}/${setNumber}`,
    ADD_SANCTION: (uniqueCode) => `${BACK_IP}/referee/add-sanction/${uniqueCode}`
  },
  OPEN: {
    // Endpoints públicos
  }
}
```

6.2 WebSocket (Real-Time Communication)

Configuration:

```
const socket = new SockJS(`${BACK_IP}/ws`);
const stompClient = Stomp.over(socket);

// Subscription to specific game topic
stompClient.subscribe(`/topic/games/${uniqueCode}`, (message) => {
  const updatedGame = JSON.parse(message.body);
  // Update game state
});
```

Real-Time Events:

- Score updates
- Set changes
- Sanctions recorded
- Game start/finish

Implementation Details:

- **Connection**: Automatic connection establishment on component mount
- **Reconnection**: Automatic reconnection on connection loss
- **Topics**: Game-specific topics with unique code identification
- **Message Format**: JSON serialized game objects
- **Error Handling**: Connection error detection and user notification
- **Cleanup**: Proper disconnection on component unmount

Features:

- Persistent connection management
- Automatic subscription handling
- Real-time game state synchronization
- Bi-directional communication for referee actions
- Live score broadcasting to all connected clients

7. Estado y Hooks

7.1 Hooks Personalizados

useWebSocket

```
useGameWebSocket(uniqueCode, onUpdate);
```

Funcionalidades:

- Conexión automática al WebSocket
- Suscripción a topic específico
- Recepción de actualizaciones en tiempo real
- Envío de mensajes al servidor
- Reconexión automática

7.2 Gestión de Estado

Estado Local (useState):

- Formularios
- Modales
- Loading states
- Listas y filtros

Sesión (localStorage):

- Token de autenticación
- Datos de usuario

8. Estilos y UI

8.1 Sistema de Colores

Definido en `css/colors.css`:

```
:root {
  --primary-color: #...;
  --secondary-color: #...;
  --accent-color: #...;
  --background-color: #...;
  --text-color: #...;
  --error-color: #...;
  --success-color: #...;
}
```

8.2 Diseño Responsivo

Breakpoints:

- Mobile: < 768px
- Tablet: 768px - 1024px
- Desktop: > 1024px

Resolución Mínima para Admin:

- Panel de administración requiere resolución mínima
- Componente `SRE.jsx` muestra error si no cumple

9. Seguridad

9.1 Autenticación

- **JWT Tokens**: Access y Refresh tokens
- **Validación de Sesión**: En cada carga de página
- **Expiración**: Tokens con tiempo de vida limitado

9.2 Protección de Rutas

- **Guards**: Verificación de autenticación y roles
- **Redirección**: Automática según permisos
- **Validación Backend**: Doble validación en servidor

9.3 Validación de Formularios

- Validación en cliente (React)
- Validación en servidor (Backend)
- Mensajes de error claros
- Prevención de XSS

10. Flujos Principales

10.1 Flujo de Login

1. Usuario ingresa credenciales
2. `auth.mjs::login()` envía request a backend
3. Backend valida y retorna tokens
4. Tokens se guardan en cookies
5. Sesión se guarda en localStorage

10.2 Flujo de Registro

1. Usuario completa formulario
2. `auth.mjs::register()` envía datos
3. Backend crea usuario y envía email de verificación
4. Usuario hace clic en link del email
5. `ValidateAccount.jsx` verifica token
6. Cuenta activada → Redirección a login

10.3 Flujo de Arbitraje

1. Árbitro selecciona partido de su lista
2. Sistema carga datos del partido
3. WebSocket establece conexión
4. Árbitro inicia el partido
5. Backend crea primer set automáticamente
6. Árbitro anota puntos en tiempo real
7. Cambios se propagan vía WebSocket
8. Espectadores ven actualizaciones en vivo
9. Árbitro finaliza el partido
10. Sistema calcula y guarda resultado final

10.4 Flujo de Administración

1. Admin accede al panel
2. Selecciona entidad a gestionar
3. Lista se carga desde backend
4. Admin puede:
    - Crear nueva entidad
    - Editar existente
    - Eliminar (con confirmación)
    - Buscar/filtrar
5. Cambios se sincronizan con backend
6. Validaciones en ambos lados

Conclusión

El frontend de DOT & DOT implementa una arquitectura moderna basada en React 19, con comunicación en tiempo real mediante WebSockets, autenticación segura mediante JWT, y una interfaz de usuario intuitiva que proporciona una experiencia fluida en la gestión de competiciones de voleibol.

Tecnologías Utilizadas

- **React 19** - Framework principal
- **Vite** - Build tool y dev server
- **React Router DOM v7** - Routing
- **React Hooks** - Gestión de estado (useState, useEffect, useContext)
- **WebSocket + STOMP** - Comunicación en tiempo real
- **Fetch API** - Peticiones HTTP
- **CSS Modules** - Estilos encapsulados
- **react-simple-maps** - Mapas interactivos

Características Implementadas

1. Sistema completo de autenticación con verificación por email
2. Guards de rutas basados en roles (Admin, Referee, User)
3. Panel de administración con CRUD completo de todas las entidades
4. Panel de árbitro con gestión de partidos en tiempo real
5. Visualización en vivo de partidos mediante WebSockets
6. Mapa interactivo de clubes de Extremadura con geolocalización
7. Sistema de favoritos con sincronización en backend
8. Consulta pública de competiciones, clasificaciones y partidos
9. Perfil de usuario con gestión de datos personales
10. Arquitectura modular (Pages-Components-Services)
11. Validaciones en cliente y servidor
12. Diseño responsive para móvil, tablet y desktop
13. Feedback visual con estados de carga y errores

Escalabilidad y Mantenimiento

La arquitectura implementada permite:

- Separación de responsabilidades por módulos
- Fácil extensión con nuevos componentes y rutas
- Testing independiente por capas
- Reutilización de componentes y servicios
- Integración con nuevas APIs externas