Documentación tecnica del Proyecto

1. **Introducción**

- **Nombre del Proyecto**: Proyecto Gestiones Principales de la red social de motos con API Mock
- **Descripción**: Llevar a cabo las funcionalidades de login, alta y baja de los clubes con Ayax y usando API Mock.
- **Objetivo**: Investigar por nuestra cuenta el funcionamiento de Ayax y de la API.

2. Roles del Equipo y Contribuciones

- Eduardo Ballesteros: Jefe de equipo y encargado de implementar las distintas partes del proyecto y investigación de Ayax.
- Ruben Pacho: Encargado de la investigación y implemetación de Ayax en el proyecto.
- Sergio Alfonseca: Encargado de la investigación y del funcionamiento de la API y de explicar los conocimientos a Ruben y a Eduardo
- Carlos Haro: Encargado del diseño de la página web y de la investigación y funcionamiento de la API.

3. Requisitos previos

Para desarollar y ejecutar el proyecto hemos tenido que contar con las siguientes herramientas y dependencias:

- Node: Requerido para usar npm y ejecutar el JSON, la herramienta que hemos usado para simular nuestra API.
- JSON Server: Permite simular una API para el proyecto, utilizando una archivo db.json que suple a la base de datos.
- Eclipse IDE: IDE en el que se ha desarollado el proyecto con la estructura de Dynamic Web Proyect.

4. **Estructura del Proyecto**

Dynamic Web Project

- Este es el proyecto principal de tu aplicación web, donde se desarrolla el HTML, CSS y JavaScript, y donde se configura la conexión AJAX a la API.

- **Carpetas y Archivos Recomendados**:

  - **WebApp**: Contiene los archivos visibles en el frontend.
    - index.html: Página principal (por ejemplo, página de login).
    - login.html: Página de alta de usuarios.
    - delete.html: Página de baja de usuarios.
    - **css/**: Carpeta que incluye los estilos CSS de los distintos html.

    - **js/**: Carpeta para el JavaScript de la aplicación, incluyendo:
        - server.js: Archivo JavaScript para gestionar las solicitudes AJAX a la API.
        - script.js: Archivo que genera algunas funcionalidades para los botones.
    - **img/** (opcional): Carpeta para imágenes.

  - **ApiMock**
    - **db.json**: Archivo JSON que funciona como "base de datos" de prueba para la API mock.
  -

- Ejemplo de contenido:


```
[
  {
    "nombre_club": "asdf",
    "colores_club": "asdf",
    "mail_club": "asdf",
    "contraseña_club": "asdf"
  },
  {
    "nombre_club": "a",
    "colores_club": "a",
    "mail_club": "a",
    "contraseña_club": "d"
  },
```


5. **Descripción de AJAX y Funcionalidad**

El propósito de implementar AJAX en este proyecto es permitir la comunicación asíncrona entre el cliente (interfaz de usuario) y el servidor sin necesidad de recargar la página. Esto mejora la experiencia del usuario al hacer que las interacciones sean más rápidas y fluidas, ya que solo se actualizan los datos necesarios sin afectar el contenido de toda la página.

- Método Alta

```
function registerClub() {
    const nombreClub = document.getElementById('nombre_club').value;
    const coloresClub = document.getElementById('colores_club').value;
    const emailClub = document.getElementById('mail_club').value;
    const passwordClub = document.getElementById('contraseña_club').value;
    const confirmPassword = document.getElementById('confirmar_contraseña').value;

    if (passwordClub !== confirmPassword) {
        alert("Las contraseñas no coinciden.");
        return;
    }

    const data = {
        nombre_club: nombreClub,
        colores_club: coloresClub,
        mail_club: emailClub,
        contraseña_club: passwordClub
    };

    fetch('http://localhost:3000/api/register', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify(data)
    })
    .then(response => response.json())
    .then(data => {
        if (data.error) {
            alert(data.error);
        } else {
            alert(data.message); // Club registrado exitosamente
        }
    })
    .catch(error => console.error('Error:', error));
}
```

- Método Baja

```
function deleteClub() {
    const emailClub = document.getElementById('mail_club').value;

    const data = {
        mail_club: emailClub
    };

    fetch('http://localhost:3000/api/delete', {
        method: 'DELETE',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify(data)
    })
    .then(response => response.json())
    .then(data => {
        if (data.error) {
            alert(data.error); // Error al eliminar
        } else {
            alert(data.message); // Club eliminado exitosamente
            // Aquí puedes limpiar los campos o actualizar la interfaz
        }
    })
    .catch(error => console.error('Error:', error));
}
```

- Método login

```
function loginClub() {

    const emailClub = document.getElementById('mail_club_login').value;
    const passwordClub = document.getElementById('contraseña_club_login').value;

    const data = {
        mail_club: emailClub,
        contraseña_club: passwordClub
    };

    fetch('http://localhost:3000/api/login', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify(data)
    })
    .then(response => response.json())
    .then(data => {
        if (data.error) {
            alert(data.error); // Error de inicio de sesión
        } else {
            alert(data.message); // Inicio de sesión exitoso
            window.location.href = "delete.html";
            // Aquí puedes redirigir al usuario o actualizar la interfaz
        }
    })
    .catch(error => console.error('Error:', error));
}
```