

# **TECNOLOGÍAS DE DESARROLLO EN EL CLIENTE**



**ITESO, Universidad  
Jesuita de Guadalajara**

**Definición de Proyecto Integrador:  
NFL Fantasy**

**Hecho por:**  
**Jesus Eduardo Castellanos Pelayo 739449**  
**Carlos Emiliano Rodríguez Núñez 738288**

**27/11/2024**

## **Descripción:**

Nuestro proyecto consiste en el desarrollo de una plataforma interactiva como Fantasy NFL, donde los usuarios podrán crear y gestionar sus propios equipos de fútbol americano basados en jugadores reales de la NFL. La página incluirá funcionalidades como:

- **Creación de ligas privadas y públicas:** Los usuarios podrán unirse o crear ligas con sus amigos o con otros participantes de la comunidad.
- **Draft en vivo:** Los jugadores seleccionarán a sus atletas en un proceso de selección en tiempo real antes o durante la temporada.
- **Gestión semanal de alineaciones:** Los usuarios podrán ajustar su equipo titular semana a semana, alineando a los jugadores según el calendario de la NFL.
- **Puntuación basada en estadísticas reales:** Los jugadores obtendrán puntos basados en el desempeño de sus atletas en los partidos reales de la NFL.

El objetivo es crear una plataforma que imite la experiencia del popular sitio de Fantasy NFL.

Aunque una plataforma de este tipo ya existe creemos que es buena idea desarrollar algo como esto ya que nos pareció una gran idea para poder probar verdaderamente todo lo que hemos aprendido en el curso. De igual manera, agregaremos características que nos parecen importantes para una página de este tipo.

## **Stack de Tecnologías a utilizar:**

Frontend

1. HTML5 / CSS3 / SASS
2. JavaScript / TypeScript
3. Angular
4. Bootstrap

Backend

1. Node.js
2. Express
3. Mongoose
4. Socket.io

## **Autentificación**

1. MongoDB
2. GoogleAuth

## **Base de Datos**

1. MongoDB

### **Roles de usuarios:**

#### 1. Administrador del Sistema

Supervisa toda la plataforma. Tiene acceso completo y es responsable de la gestión de usuarios, ligas y equipos.

- **Permisos:**

- Gestionar usuarios (crear, editar, eliminar).
- Moderar ligas (crear, editar, eliminar ligas).
- Ver todas las estadísticas y resultados.
- Manejar configuraciones del sistema (puntajes, reglas de juego, etc.).
- Gestionar cualquier contenido inapropiado o reportado.
- Hacer cualquier cosa que puedan hacer los demás tipos de usuarios

#### 2. Jugador que crea una liga

Usuario que organiza una liga específica. Tiene control sobre las reglas y el funcionamiento de su liga particular.

- **Permisos:**

- Crear y gestionar una liga (nombre, reglas, número de equipos).
- Invitar o expulsar jugadores de la liga.
- Configurar el draft (tipo, orden, tiempo, etc.).
- Modificar reglas de puntuación para su liga (si es permitido).
- Desempatar disputas o conflictos en su liga.
- Gestionar el calendario de enfrentamientos.

#### 3. Jugador que juega en una liga

Usuario que participa en una o más ligas, gestionando su equipo y compitiendo semanalmente.

- **Permisos:**

- Unirse a ligas (públicas o por invitación).
- Participar en el draft para seleccionar jugadores.
- Configurar su equipo semana a semana (alinear jugadores, hacer cambios).
- Realizar intercambios con otros jugadores de la liga.
- Ver estadísticas de sus jugadores y rivales.
- Ver el calendario y los enfrentamientos.
- Participar en chats o foros de la liga.

#### 4. Espectador

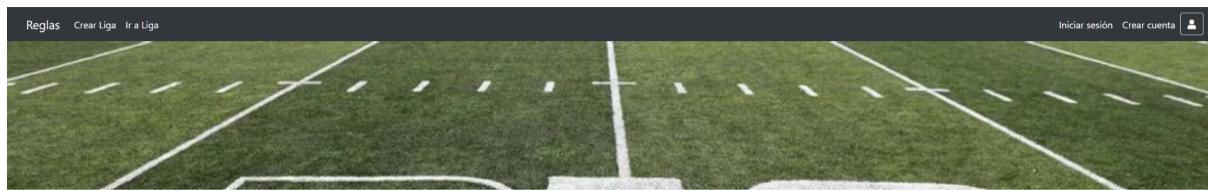
Usuario no registrado que puede acceder a contenido limitado, como ver estadísticas globales, pero sin la capacidad de participar en ligas.

- **Permisos:**

- Ver estadísticas generales de jugadores y equipos.
- Explorar ligas públicas (sin unirse).
- Consultar reglas generales del juego.
- Seguir resultados y clasificaciones, pero sin acceso a interacción o personalización.

## Vistas

## Vista de inicio:



### Jugadores Destacados

Christian McCaffrey Equipo: San Francisco 49ers Edad: 25 Descripción: Uno de los corredores más versátiles de la NFL.	Dak Prescott Equipo: Dallas Cowboys Edad: 28 Descripción: Líder y mariscal de campo estrella de los Cowboys.	Aaron Rodgers Equipo: Green Bay Packers Edad: 37 Descripción: Uno de los mariscos de campo más talentosos de la historia.	Justin Fields Equipo: Chicago Bears Edad: 22 Descripción: Prometedor mariscal de campo con gran potencial.
--	---	--	---



### ¿Qué es Fantasy Football?

Fantasy Football es un juego en el que los participantes actúan como gerentes generales de equipos virtuales de fútbol americano profesional. Los equipos compiten basándose en las estadísticas de los jugadores en la vida real.

### ¿Cómo jugar Fantasy Football?

Para jugar Fantasy Football, primero debes unirte o crear una liga. Luego, seleccionas jugadores en un draft para formar tu equipo. Cada semana, tus jugadores ganan puntos basados en su rendimiento en los partidos reales. El objetivo es acumular más puntos que tus oponentes.

### Power Rankings de la Semana



Kansas City  
Chiefs  
Rank 1

Buffalo Bills  
Rank 2

Minnesota Vikings  
Rank 3

Seattle Seahawks  
Rank 4

## Vista de registro:

Reglas Crear Liga Ir a Liga

Iniciar sesión Crear cuenta

## Regístrate en NFL Fantasy

Crea tu equipo y compite con amigos y otros fanáticos.

**Crear Cuenta**

Nombre Completo  
Ingresá tu nombre completo

Correo Electrónico  
Ingresá tu correo electrónico

Contraseña  
Ingresá una contraseña segura

Confirmar Contraseña  
Confirma tu contraseña

**Registrarse**

---

Power Rankings de la Semana

Kansas City Chiefs  
Rank 1

Buffalo Bills  
Rank 2

Minnesota Vikings  
Rank 3

Seattle Seahawks  
Rank 4

### Vista de login:

Reglas Crear Liga Ir a Liga

Iniciar sesión Crear cuenta

## Inicia Sesión en NFL Fantasy

**Inicio de Sesión**

Correo Electrónico  
Ingresá tu correo electrónico

Contraseña  
Ingresá tu contraseña

**Iniciar Sesión**

**Iniciar Sesión Admin**

---

Power Rankings de la Semana

Kansas City Chiefs  
Rank 1

Buffalo Bills  
Rank 2

Minnesota Vikings  
Rank 3

Seattle Seahawks  
Rank 4

### Vista de chat:

NFL Fantasy Reglas Crear Liga Ir a mi Liga Chat Iniciar sesión Crear cuenta

Welcome, Eduardo

Type your message here... Enviar Anadir archivo

---

Power Rankings de la Semana

Rank	Team
1	KC Chiefs
2	Buffalo Bills
3	Minnesota Vikings
4	Seattle Seahawks

## Vista de my-account:

NFL Fantasy Reglas Crear Liga Ir a mi Liga Chat Bienvenido, Usuario Cerrar sesión

Tu Cuenta

Nombre: Eduardo Castellanos  
Email: jesus.castellanos@iteso.mx

Token de sesión:  
eyJhbGciOiJIzIifNisInhR5cD6ikpXWCI9eyJpZC06jy3NDcGMGU4ZmQzZW2lZTU0MzhMzhHYSlsmlhdC06MtcaMjc0MzU2NCwZXhwIjoxNzMyNzQ3MTY0fQxfNM8964h4nJS18pU-JPvbd3MfYha6nOSVpkDybbE

Cerrar sesión

---

Power Rankings de la Semana

Rank	Team
1	KC Chiefs
2	Buffalo Bills
3	Minnesota Vikings
4	Seattle Seahawks

## Vista de liga-beta:

NFL Fantasy Reglas Crear Liga Ir a mi Liga Chat Bienvenido, Usuario Cerrar sesión

Gestiona tus Ligas

Buscar Liga

Nombre de la Liga  
Ingresá el nombre de la liga  
Buscar Liga

Crear Liga

Nombre de la Liga  
Ingresá el nombre de la liga  
Agregar Jugadores  
Ingresá los nombres de los jugadores, separados por comas  
Crear Liga

---

Power Rankings de la Semana

Rank	Team
1	KC Chiefs
2	Buffalo Bills
3	Minnesota Vikings
4	Seattle Seahawks

## Vista de play-beta:

The screenshot shows a comparison between two teams. On the left, under 'Mi equipo' (My team), there is a placeholder box labeled 'Mis Jugadores'. On the right, under 'Equipo enemigo' (Opponent team), there is a box titled 'Jugadores del Contrincante' containing three players: Rodgers (QB), Diggs (WR), and Nick Chubb (RB). Below these boxes, a summary box displays 'Mis Puntos: 0' and 'Puntos del Contrincante: 0 por manco'. A blue 'Guardar equipo' button is at the bottom. At the very bottom, there's a section titled 'Draftear' (Draft) with three player cards: Aaron Rodgers (QB), Lamar Jackson (QB), and Davante Adams (WR), each with an 'Agregar' (Add) button.

## Vista de reglas:

The screenshot shows the 'Fantasy Football' rules section. It includes a question '¿Qué es Fantasy Football?' with a detailed answer about the game being a managerial sport based on real-life player statistics. It also includes a section titled 'Reglas Básicas del Fantasy Football' with a list of basic rules and a question '¿Cómo jugar Fantasy Football?' with a brief explanation of the process.

**¿Qué es Fantasy Football?**

Fantasy Football es un juego en el que los participantes actúan como gerentes generales de equipos virtuales de fútbol americano profesional. Los equipos compiten basándose en las estadísticas de los jugadores en la vida real.

**Reglas Básicas del Fantasy Football**

- Cada liga tiene entre 8 y 14 equipos.
- El draft es donde seleccionas jugadores para tu equipo; puede ser en vivo o automatizado.
- Los jugadores ganan puntos basados en su rendimiento en partidos reales.
- Cada semana debes alinear titulares y suplentes.
- El equipo con más puntos al final de la temporada regular y los playoffs gana la liga.

**¿Cómo jugar Fantasy Football?**

Para jugar Fantasy Football, primero debes unirte o crear una liga. Luego, seleccionas jugadores en un draft para formar tu equipo. Cada semana, tus jugadores ganan puntos basados en su rendimiento en los partidos reales. El objetivo es acumular más puntos que tus oponentes.

## Descripción de vistas

**Footer y header:** Estos son incluidos en todas las vistas, en el header se puede ir a la sección de las reglas, creación de una liga y unirse a una liga, además de crear una cuenta, iniciar sesión y ver datos de tu perfil.

En el footer se pueden ver los power rankings de cada semana además del rank en el que se encuentra cada equipo

Vista de inicio: En esta vista se puede ver información de los 4 jugadores más destacados actualmente, dentro de lo que se muestra está incluida su edad, equipo y una breve descripción de qué es lo que los hace destacar. Seguido de esto hay una sección donde puedes iniciar sesión o crear una liga para poder jugar a NFL Fantasy, además de una breve descripción de que es NFL Fantasy y como jugarlo.

Vista de registro: Aquí hay un formulario en donde se te pide ingresar tu nombre completo, correo electrónico, contraseña y confirmar tu contraseña para así poder crear tu cuenta.

Vista de login: Aquí hay un formulario en donde se te pide ingresar tu correo electrónico y contraseña para así poder iniciar tu sesión.

Vista de chat: Aquí se puede hablar con otros usuarios conectados a nuestra plataforma, además se pueden enviar archivos como por ejemplo myTeam.json que es un archivo con toda la información de un equipo creada por un usuario

Vista de my-account: Aquí se puede ver información de la cuenta como nombre, correo y token.

Vista de liga-beta: Aquí se puede crear una liga o unirse a una liga, para unirse a una simplemente es necesario escribir el nombre de la liga en un formulario, en el caso de querer crear una es necesario escribir el nombre de la liga y el nombre de los jugadores que estarán en tu equipo, estos deberán de ir separados por comas.

Vista de play-beta: Aquí se puede ver tu equipo y el de tu rival, además de los puntos que tiene cada uno de los equipos y los distintos jugadores que pueden ser agregados al equipo.

Vista de reglas: Aquí se pueden leer las reglas de la página además de una breve descripción de lo que ofrece.

### **Envío de archivos:**

# Welcome, Eduardo

Eduardo 11/27/2024, 3:51:31 PM

Archivo añadido.

[Descargar archivo: 1732744291445-server.js](#)

Eduardo undefined

undefined

[Descargar archivo: 1732744291445-server.js](#)

Type your message here...

Enviar

Añadir archivo

# Welcome, Eduardo

Eduardo 11/27/2024, 3:51:31 PM

Archivo añadido.

[Descargar archivo: 1732744291445-server.js](#)

Eduardo undefined

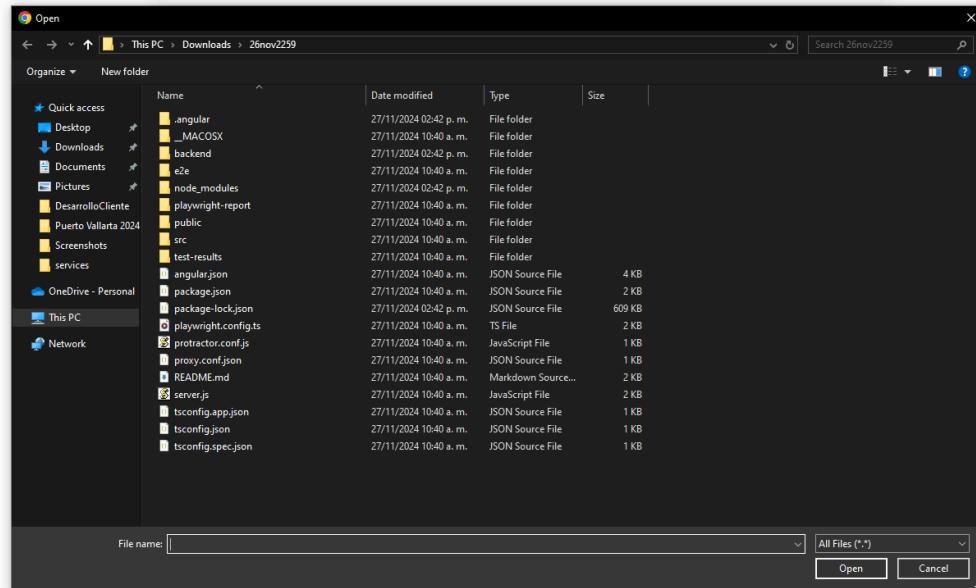
undefined

[Descargar archivo: 1732744291445-server.js](#)

Type your message here...

Enviar

Añadir archivo



## Uso de sockets:

### Welcome, Juanito

**Juanito** 27/11/2024, 3:52:52 p.m.

hola, quien esta aqui?

Type your message here...

Enviar

Añadir archivo

### Welcome, Eduardo

**Eduardo** 11/27/2024, 3:51:31 PM

Archivo añadido.

[Descargar archivo: 1732744291445-server.js](#)

**Eduardo** undefined

undefined

[Descargar archivo: 1732744291445-server.js](#)

**Juanito** 27/11/2024, 3:52:52 p.m.

hola, quien esta aqui?

**Eduardo** 27/11/2024, 3:53:19 p.m.

Yo

Type your message here...

Enviar

Añadir archivo

## Capturas de código para sockets:

```
src > app > services > ts chat.service.ts > ChatService
1 // src/app/services/chat.service.ts
2 import { Injectable } from '@angular/core';
3 import { io, Socket } from 'socket.io-client';
4
5 @Injectable({
6 providedIn: 'root'
7 })
8 export class ChatService {
9   private socket: Socket | null = null;
10  private socketUrl = 'http://localhost:5002';
11
12  connect(): void {
13    if (!this.socket) {
14      this.socket = io(this.socketUrl);
15      this.socket.on('connect', () => {
16        console.log('Conectado al servidor de Socket.IO');
17      });
18
19      this.socket.on('connect_error', (error) => {
20        console.error('Error de conexión:', error);
21      });
22    }
23  }
24
25  sendMessage(message: string): void {
26    if (this.socket && this.socket.connected) {
27      this.socket.emit('newMessage', { user: 'Usuario', message });
28      console.log('Mensaje enviado al servidor:', message);
29    } else {
30      console.warn('No se pudo enviar el mensaje: WebSocket no conectado.');
31    }
32  }
33
34  disconnect(): void {
35    if (this.socket) {
36      this.socket.disconnect();
37      console.log('Conexión de Socket.IO cerrada');
38    }
39  }
40}
```

```
src > app > components > layout > chat > ts chat.component.ts > ChatComponent
1 // src/app/components/layout/chat/chat.component.ts
2 import { Component, OnInit, OnDestroy } from '@angular/core';
3 import { io, Socket } from 'socket.io-client';
4 import { CommonModule } from '@angular/common';
5 import { FormsModule } from '@angular/forms';
6
7 interface ChatMessage {
8   user: string;
9   message: string;
10  date: string;
11  filePath?: string; // Ruta al archivo en el servidor
12  fileName?: string; // Nombre del archivo
13 }
14
15 @Component({
16   selector: 'app-chat',
17   standalone: true,
18   imports: [CommonModule, FormsModule],
19   templateUrl: './chat.component.html',
20   styleUrls: ['./chat.component.scss']
21 })
22 export class ChatComponent implements OnInit, OnDestroy {
23   public newMessage: string = '';
24   public messages: ChatMessage[] = [];
25   private userName: string = '';
26   private socket: Socket;
27
28   constructor() {
29     this.socket = io('http://localhost:5002'); // Conectar al servidor de Socket.IO
30   }
31
32   ngOnInit(): void {
33     this.requestNotificationPermission();
34     this.getUserName();
35     this.initSocketEventListeners();
36     this.socket.emit('newUser', { user: this.userName });
37   }
38
39   requestNotificationPermission(): void {
40     if ('Notification' in window && Notification.permission !== 'granted') {
41       Notification.requestPermission().then((permission) => {
42         if (permission === 'granted') {
43           console.log('Permiso para notificaciones concedido.');
44         } else {
45           console.warn('Permiso para notificaciones denegado.');
46         }
47       });
48     }
49   }
50 }
```

```

50
51   getUserName(): void {
52     this.userName = localStorage.getItem('username') || '';
53     if (!this.userName) {
54       this.userName = prompt("Ingrese su nombre de usuario:") || 'Invitado';
55       localStorage.setItem('username', this.userName);
56     }
57     document.getElementById('user').innerHTML = this.userName;
58   }
59
60   initSocketEventListeners(): void {
61     this.socket.on('message', (msg: string) => {
62       console.log(msg);
63     });
64
65     this.socket.on('newMessage', (data: ChatMessage) => {
66       console.log(`Mensaje recibido de ${data.user}: ${data.message}`);
67       this.messages.push(data);
68       this.updateMessages();
69     });
70
71     this.socket.on('fileMessage', (data: ChatMessage) => {
72       console.log(`Archivo recibido de ${data.user}: ${data.fileName}`);
73       this.messages.push(data);
74       this.updateMessages();
75       this.showFileNotification(data); // Mostrar la notificación al recibir un archivo
76     });
77
78     this.socket.on('disconnect', () => {
79       console.log('Desconectado del servidor de Socket.IO');
80     });
81   }
82
83   /**
84    * Mostrar una notificación de archivo recibido
85    * @param data Información del archivo recibido
86    */
87   showFileNotification(data: ChatMessage): void {
88     if (Notification.permission === "granted") {
89       const notification = new Notification("Nuevo archivo recibido",
90         {
91           body: `Archivo de ${data.user}: ${data.fileName}`,
92           icon: "/assets/file-icon.png" // Cambia este ícono por uno adecuado
93         });
94
95       notification.onclick = () => {
96         window.open(data.filePath, '_blank'); // Descargar el archivo al hacer clic en la notificación
97       };
98     } else {
99       console.warn("Permiso para notificaciones no otorgado.");
100    }
101  }
102
103  sendMessage(event: Event): void {
104   event.preventDefault(); // Prevenir el comportamiento predeterminado del formulario
105   console.log('Envviando mensaje:', this.newMessage); // Verificar que se envía el mensaje
106   if (this.newMessage.trim()) {
107     this.socket.emit('newMessage', { user: this.userName, message: this.newMessage });
108     this.newMessage = '';
109   } else {
110     console.warn('No se pudo enviar el mensaje: El mensaje está vacío.');
111   }
112 }
113
114 triggerFileInput(): void {
115   const fileInput = document.getElementById('fileInput') as HTMLInputElement;
116   fileInput.click();
117 }
118
119 handleFileInput(event: Event): void {
120   const input = event.target as HTMLInputElement;
121   if (input.files && input.files.length > 0) {
122     const file = input.files[0];
123     const formData = new FormData();
124     formData.append('file', file);
125
126     fetch('http://localhost:5002/upload', {
127       method: 'POST',
128       body: formData
129     })
130     .then(response => response.json())
131     .then(data => {
132       if (data.path) {
133         console.log('Archivo subido exitosamente:', data.filename);
134         // Emitir un evento de archivo al servidor
135         this.socket.emit('fileMessage', { path: data.path, filename: data.filename });
136         // Añadir el archivo como un mensaje en el cliente
137         // ...
138     })
139   }
140 }

```

```

135     // Añadir el archivo como un mensaje en el cliente
136     this.messages.push({
137       user: this.userName,
138       message: 'Archivo añadido.',
139       date: new Date().toLocaleString(),
140       filePath: data.path,
141       fileName: data.filename
142     });
143     this.updateMessages();
144   }
145 )
146 .catch(error => console.error('Error al cargar el archivo:', error));
147
148 }
149
150 updateMessages(): void {
151   const messagesContainer = document.getElementById('messages');
152   if (messagesContainer) {
153     messagesContainer.innerHTML = '';
154     this.messages.forEach((msg) => {
155       const p = document.createElement('p');
156       p.innerHTML = `<strong>${msg.user}</strong> <small>${msg.date}</small><br>${msg.message}`;
157
158       if (msg.filePath) {
159         const a = document.createElement('a');
160         a.href = msg.filePath;
161         a.innerText = `Descargar archivo: ${msg.fileName}`;
162         a.target = '_blank';
163         p.appendChild(document.createElement('br'));
164         p.appendChild(a);
165       }
166
167       messagesContainer.appendChild(p);
168     });
169   }
170 }
171
172 ngOnDestroy(): void {
173   this.socket.disconnect();
174 }
175 }

```

```

JS server.js > ...
1 // server.js
2
3 const express = require('express');
4 const http = require('http');
5 const socketio = require('socket.io');
6 const multer = require('multer');
7 const path = require('path');
8
9 const app = express();
10 const server = http.createServer(app);
11 const io = socketio(server, {
12   cors: {
13     origin: "http://localhost:4200",
14     methods: ["GET", "POST"]
15   }
16 });
17 const cors = require('cors');
18 app.use(cors());
19 // Configuración de multer para almacenamiento de archivos
20 const storage = multer.diskStorage({
21   destination: (req, file, cb) => {
22     cb(null, 'public/uploads/'); // Carpeta de destino para los archivos
23   },
24   filename: (req, file, cb) => {
25     cb(null, `${Date.now()}-${file.originalname}`); // Nombre del archivo
26   }
27 });
28 const upload = multer({ storage: storage });
29
30 app.use(express.static('public'));
31
32 // Ruta para cargar archivos
33 app.post('/upload', upload.single('file'), (req, res) => {
34   if (req.file) {
35     console.log(`Archivo añadido: ${req.file.filename}, guardado en /uploads/`);
36     res.status(200).json({ filename: req.file.filename, path: `/uploads/${req.file.filename}` });
37   } else {
38     res.status(400).json({ error: 'No file uploaded' });
39   }
40 });
41
42 io.on('connection', (socket) => {
43   console.log('Nuevo usuario conectado');
44
45   socket.on('newUser', (data) => {

```

```

42  io.on('connection', (socket) => {
43    console.log('Nuevo usuario conectado');
44
45    socket.on('newUser', (data) => {
46      socket.username = data.user;
47      socket.broadcast.emit('message', `${data.user} se ha unido a la conversación`);
48    });
49
50    socket.on('newMessage', (msg) => {
51      const date = new Date().toLocaleString();
52      io.emit('newMessage', { user: socket.username, message: msg.message, date });
53    });
54
55    socket.on('fileMessage', (fileData) => {
56      io.emit('fileMessage', { user: socket.username, filePath: fileData.path, fileName: fileData.filename });
57    });
58
59    socket.on('disconnect', () => {
60      socket.broadcast.emit('message', `${socket.username} ha dejado la conversación`);
61    });
62  });
63
64  const PORT = process.env.PORT || 5002;
65  server.listen(PORT, () => {
66    console.log(`Corriendo en el puerto ${PORT}!`);
67  });

```

## Código de pruebas automatizadas:

### auth.service.spec.ts:

```

import { TestBed } from '@angular/core/testing';
import { HttpClientTestingModule } from '@angular/common/http/testing';
import { AuthService } from './auth.service';

describe('AuthService', () => {
  let service: AuthService;

  beforeEach(() => {
    TestBed.configureTestingModule({
      imports: [HttpClientTestingModule], // Importa el módulo de pruebas para HttpClient
      providers: [AuthService],
    });
    service = TestBed.inject(AuthService);
  });

  it('should return true for valid login credentials', () => {
    const result = service.login('salamaleco@hotmail.com', 'Hola12345');

    expect(result).toBeTruthy();
  });

  it('should return true for valid registration data', () => {

```

```
const result = service.register('Test User', 'test@example.com', 'password123');

expect(result).toBeTruthy();
});

});
```

```
import { TestBed, ComponentFixture } from '@angular/core/testing';
import { HttpClientTestingModule } from '@angular/common/http/testing';
import { AccountComponent } from './account.component';
import { AuthService } from '../../../../../services/auth.service';

describe('AccountComponent', () => {
  let component: AccountComponent;
  let fixture: ComponentFixture<AccountComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      imports: [HttpClientTestingModule],
      providers: [AuthService],
    }).compileComponents();

    fixture = TestBed.createComponent(AccountComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create the component', () => {
    expect(component).toBeTruthy();
  });

  it('should call AuthService.register with correct arguments on form submit', () => {
    const authService = TestBed.inject(AuthService);
    spyOn(authService, 'register').and.returnValue(true); // Espiar el método register y simular que devuelve true

    // Simula el llenado del formulario
    component.registerForm.setValue({
      name: 'Test User',
      email: 'test@example.com',
    });
  });
});
```

```

    password: 'password123',
    confirmPassword: 'password123'
});

component.onSubmit();

expect(authService.register).toHaveBeenCalledOnceWith('Test User',
'test@example.com', 'password123');
};

});

```

```

import { test, expect } from '@playwright/test';

test('Login test', async ({ page }) => {
    // Navega a la página de inicio de sesión
    await page.goto('http://localhost:4200/login');

    // Rellenar los campos de email y contraseña
    await page.fill('input[id="email"]', 'salamaleco@hotmail.com');
    await page.fill('input[id="password"]', 'Hola12345');

    // Hacer clic en el botón de login
    await page.click('button[type="submit"]');

    // Esperar a que la navegación ocurra después del login
    await page.waitForNavigation();

    // Verificar que la página redirige correctamente al home (o cualquier otra comprobación)
    await expect(page).toHaveURL('http://localhost:4200/home');
});

```

**Capturas de pruebas automatizadas:**

Karma v 6.4.0 - connected; test: complete;

Chrome 131.0.0.0 (Windows 10) is idle

Jasmine 4.6.1

• • • • • • • • • •

13 specs, 0 failures, randomized with seed 69244

finished in 10.488s

```

ChatService
  • should be created
PlayBetaComponent
  • should create
AuthService
  • should be created
ReglasComponent
  • should create
ChatComponent
  • should create
AccountComponent
  • should call AuthService.register with correct arguments on form submit
  • should create the component
DragAndDropDirective
  • should create an instance
authguard
  • should be created
AuthService
  • should return true for valid registration data
  • should return true for valid login credentials
LigaBetaComponent
  • should create
HomeComponent
  • should create

```

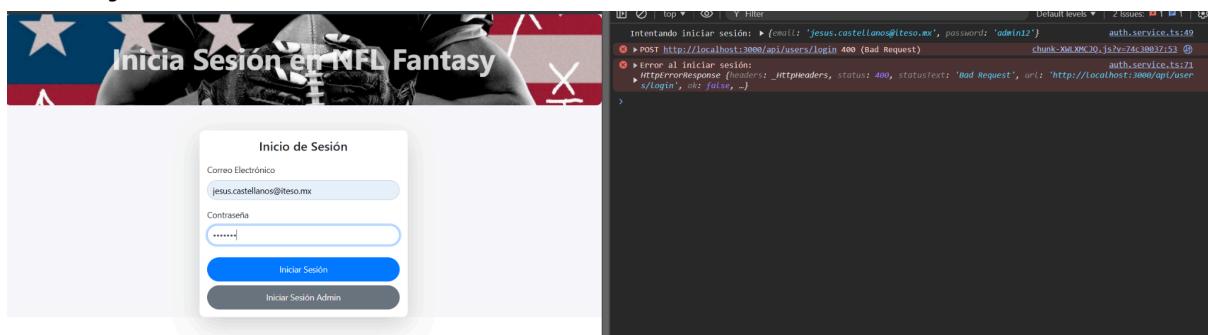
All 3 | Passed 3 | Failed 0 | Flaky 0 | Skipped 0

11/27/2024, 4:04:01 PM Total time: 7.6s

tests/login.spec.ts

- ✓ Login test [chromium] tests/login.spec.ts:3 3.9s
- ✓ Login test [firefox] tests/login.spec.ts:3 4.8s
- ✓ Login test [webkit] tests/login.spec.ts:3 2.4s

## Mensaje de error en formulario invalido:



## Conclusiones:

Este proyecto nos ayudó mucho a poder aplicar gran parte de los conocimientos adquiridos a lo largo del semestre, desde una simple creación de componentes con angular hasta poder consumir Websockets y APIs por medio de servicios.