File

`src/app/services/team.service.ts`

Description

Service that contains all the routes to our api regarding Teams

Index

Properties

- Private baseUrl

Methods

- Public addMember
- Public bestTeam
- Public changeName
- Public createTeam
- Public delete
- Public deleteMember
- Public getTeamInfo
- Public getteamWinner
- Public searchTeam

Constructor

`constructor(http: HttpClient)`

Defined in src/app/services/team.service.ts:18

Constructor in which we inject httpClient to make http requests

Parameters :

Name	Type	Optional
http	`HttpClient`	No

Methods

Public addMember

`addMember(idTeam: number, data)`

Defined in src/app/services/team.service.ts:64

Add new member (you must send the field of the bd "member_1, member_2, etc" and the hero code)

Parameters :

Name	Type	Optional
idTeam	`number`	No
data		No

Returns : `any`

Public bestTeam

`bestTeam()`

Defined in src/app/services/team.service.ts:41

Get team with higher stats

Returns : `Observable<Team>`

Team

Public changeName

`changeName(idTeam: number, team)`

Defined in src/app/services/team.service.ts:51

Change name

Parameters :

Name	Type	Optional
idTeam	number	No
team		No

Returns : any

Public createTeam

`createTeam(team)`

Defined in src/app/services/team.service.ts:29

Create a new team

Parameters :

Name	Optional
team	No

Returns : any

Public delete

`delete(idTeam: number)`

Defined in src/app/services/team.service.ts:87

Delete Team

Parameters :

Name	Type	Optional
idTeam	number	No

Returns : any

Public deleteMember

`deleteMember(idTeam: number, member)`

Defined in src/app/services/team.service.ts:76

Delete member

Parameters :

Name	Type	Optional
idTeam	number	No
member		No

Returns : any

Public getTeamInfo

`getTeamInfo(idUsu: number)`

Defined in src/app/services/team.service.ts:99

Get user Team

Parameters :

Name	Type	Optional
idUsu	number	No

Returns : Observable<Team>

team

Public getteamWinner

`getteamWinner(idteam1: number, idTeam2: number)`

Defined in src/app/services/team.service.ts:109

Get the team with the highest points

Parameters :

Name	Type	Optional
idteam1	number	No
idTeam2	number	No

Returns : Observable<Team>

team

Public searchTeam

`searchTeam(teamName: string)`

Defined in src/app/services/team.service.ts:118

Get the team by name

Parameters :

Name	Type	Optional
teamName	string	No

Returns : Observable<Team[]>

team[]

Properties

Private baseUrl

Type : string

Default value : `environment.BASE_API_URL`

Defined in src/app/services/team.service.ts:18

Variable in which we store the url of our api

```
import { Injectable } from '@angular/core';
import { environment } from "src/environments/environment";
import { HttpClient, HttpHeaders } from "@angular/common/http";
import { Team } from '../models/team';
import { Observable } from 'rxjs';

/**
 * Service that contains all the routes to our api regarding Teams
 */
@Injectable({
  providedIn: 'root'
})

export class TeamService {
  /**
  * Variable in which we store the url of our api
  */
  private baseUrl: string = environment.BASE_API_URL;

  /**
  * Constructor in which we inject httpClient to make http requests
  */
  constructor(private http: HttpClient) { }
```

```
/**
 * Create a new team
 * @param {Team} team
 */
public createTeam(team) {
  let headers = new HttpHeaders().set('Content-Type', 'application/json');
  let tokenAuth = (localStorage.getItem('token'));
  headers = headers.set("Authorization", `${tokenAuth}`);
  return this.http.post(`${this.baseUrl}/team/createTeam`, team,
    { headers: headers });
}

/**
 * Get team with higher stats
 * @returns Team
 */
public bestTeam(): Observable<Team> {
  return this.http.get<Team>(`${this.baseUrl}/team/bestTeam`);
}


/**
   * Change name
   * @param {number} idTeam
   * @param {Team} team
   */
public changeName(idTeam: number, team) {
  let params = JSON.stringify(team);
  let headers = new HttpHeaders().set('Content-Type', 'application/json');
  let tokenAuth = (localStorage.getItem('token'));
  headers = headers.set("Authorization", `${tokenAuth}`);
  return this.http.put(`${this.baseUrl}/team/chageName/${idTeam}`, params, { headers: headers });
}

/**
 * Add new member (you must send the field of the bd "member_1, member_2, etc" and the hero code)
 * @param {number} idTeam
 * @param {any} data
 */
public addMember(idTeam: number, data) {
  let headers = new HttpHeaders().set('Content-Type', 'application/json');
  let tokenAuth = (localStorage.getItem('token'));
  headers = headers.set("Authorization", `${tokenAuth}`);
  return this.http.put(`${this.baseUrl}/team/addMember/${idTeam}`, data, { headers: headers });
}

/**
  * Delete member
  * @param {number} idTeam
  * @param {string} member
  */
public deleteMember(idTeam: number, member) {
  let headers = new HttpHeaders().set('Content-Type', 'application/json');
  let tokenAuth = (localStorage.getItem('token'));
  headers = headers.set("Authorization", `${tokenAuth}`);
  return this.http.put(`${this.baseUrl}/team/deleteMember/${idTeam}`, member, { headers: headers });
}

/**
   * Delete Team
   * @param {number} idTeam
   */
public delete(idTeam: number) {
  let headers = new HttpHeaders().set('Content-Type', 'application/json');
  let tokenAuth = (localStorage.getItem('token'));
  headers = headers.set("Authorization", `${tokenAuth}`);
  return this.http.delete(`${this.baseUrl}/team/${idTeam}`, { headers: headers });
}

/**
* Get user Team
* @param {number} idUsu
* @returns team
*/
public getTeamInfo(idUsu: number): Observable<Team> {
  return this.http.get<Team>(`${this.baseUrl}/team/getTeamInfo/${idUsu}`);
}

/**
* Get the team with the highest points
* @param {number} idteam1
* @param {number} idteam2
* @returns team
*/
public getteamWinner(idteam1: number, idTeam2: number): Observable<Team> {
  return this.http.get<Team>(`${this.baseUrl}/team/getTeamWinner/${idteam1}/${idTeam2}`);
```

```
  }

  /**
   * Get the team by name
   * @param {string} teamName
   * @returns team[]
   */
  public searchTeam(teamName: string): Observable<Team[]> {
    return this.http.get<Team[]>(`${this.baseUrl}/team/searchTeam/${teamName}`);
  }

}
```

result-matching ""

No results matching ""